On-line Handwriting Recognition using Support Vector Machines and Hidden Markov Models approaches

Abdul Rahim Ahmad

UNIVERSITE DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET DES MATHEMATIQUES

Année : 2008

Thèse de Doctorat de l'Université de Nantes

Discipline : Traitement du Signal et des Images
Spécialité : Automatique et Informatique Appliquée

Présentée et soutenue publiquement par

Abdul Rahim AHMAD

le 29 Decembre 2008

à Universiti Teknologi Malaysia

« Reconnaissance de l'écriture manuscrite en-ligne par approche combinant systèmes à vastes marges et modèles de Markov cachés»

Jury

Rapporteurs	:	**Jean-Marc OGIER**	Professeur, Universite de La Rochelle
		Chee Peng LIM	Professeur, Universiti Sains Malaysia
Examinateurs	:	**Sheikh Hussain SHAIKH SALLEH**	Professeur, Universiti Teknologi Malaysia
		Patrick LE CALLET	Professeur, Universite de Nantes
		Marzuki KHALID	Professeur, Universiti Teknologi Malaysia
		Christian VIARD-GAUDI N	Professeur, Universite de Nantes

Directeur de Thèse : Christian VIARD-GAUDI N
Laboratoire : ***IRCCyN***
Co-encadrant : **Marzuki KHALID**
Laboratoire : ***CAIRO, Universite de Technologie de Malaisie***

N° ED 0366-xxx

DEDICATION

To my wife and children.

ACKNOWLEDGEMENTS

I would like to thank:

- God, for giving me interest to study, for giving me good health to be able to spend the time to study and for giving me the perseverance.

- My supervisors, past and present :
  - Prof. Marzuki Khalid (UTM).
  - Prof. Christian Viard Gaudin (EPUN).
  - Prof. Dominic Barba (EPUN).
  - Prof. Dr. Rubiyah (UTM).

- Universiti Tenaga Nasional (UNITEN) for providing me with the sponsorship for local study and later for full time 1 year study in Nantes.
- Universiti Teknologi Malaysia (UTM) the first partner university in the joint PhD program.
- Ecole Polytechnique de l'Universite de Nantes (EPUN) France, the second partner institution in the joint PhD program..
- The French Embassy of Malaysia, for providing the 12 months bourse for my study in France between 2001 to 2005.
- Professor Syed Abdul Kadir, Dean of College of Engineering, UNITEN, whom without fail, has supported my application for the leave to be away for the research study in EPUN.
- Dr. Mohd. Sharifuddin and Dr. Zainuddin, Dean and Deputy Dean of College of IT, UNITEN, similarly for supporting my application for the leave.
- Azizah, Alicia, Dr. Madan and Dr. Roslan for taking over or handling my teaching load in UNITEN while I was in France.
- My wife, Azian Muhamad Ariff for being patient and understanding about my struggle to complete the PhD.
- My children: Arifah, Afifah, Afiq and Atiqah for their patience to be without a father, while I am away in Nantes..
- Myself for believing that "I can do it".

Lastly, thanks to the Malaysia students in France who were always willing to host me when I visited them during my break from study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

$\Re$	Real Numbers
O_t	Observation at time t
$P(O \mid \lambda)$	Word likelihood
$p(x \mid C)$	Character or class likelihood
$P(\lambda \mid O)$	Word posterior probability
$P(C \mid x)$	Character or class posterior probability
$p(x)$	Probability density function (PDF) of continuous random variable x
$P(x)$	Probability of a discrete random variable x
α	Forward variable
β	Backward variable
a	Transition probability
b	Observation probability
$\varphi(x)$	Kernel Mapping
ANN	Artificial Neural Networks
RBF	Radial Basis Function
SVM	Support Vector Machine
HMM	Hidden Markov Models
MLP	Multilayer Perceptrons
MLE	Maximum Likelihood Estimation
MMI	Maximum Mutual Information

MCE	Minimum Classification Error
DP	Dynamic Programming
SEGREC	Segmentation by Recognition
SDNN	Space Displacement Neural Network
TDNN	Time Delay Neural Network
RNN	Recurrent Neural Network
INSEG	Input Space Segmentation
OUTSEG	Output Space Segmentation
PDA	Personal Digital Assistant
IRONOFF	IRESTE Online/Offline Isolated Handwritten Word Database
UNIPEN	Isolated Character Database collected by UNIPEN foundation
IRONOFF-UNIPEN	Isolated Character Database which is a mixture of IRONOFF and UNIPEN character databases.
MNIST	Isolated Handwritten Digit Database Modified from NIST database

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Handwriting is one of the most important ways of communication. It was used since the Stone Age where symbols were drawn on stones in order to express or convey some meaningful information. Later, handwriting was done using pen and paper. Handwriting was used for personal benefits like writing reminders and notes for ourselves or for business purposes such as writing letters, statements and filling up forms. Thus handwriting then was by human to human for conveying information.

The handwriting of each individual is unique because the process of handwriting is a physical process, which involves the mind, skeleton and muscles, controlled by the brain. Even so, individual handwriting could also differ, based on the mood and the state of mind of the person writing. The handwriting among the different stock of people (Europeans vs. Asian or French vs. Malaysian) are normally different, due to the conditioning and training during the period of growing up. However, even though the same stock of people has similar handwriting, it is an accepted fact that no two people have the same style of handwriting.

Initially, in a modern computer, the most important device used to interface them to human is a keyboard. As computers are becoming ubiquitous and more people are using it, a more natural interface is needed. The most likely candidates could be voice or handwriting. Voice or speech recognition capability and handwriting

recognition capability built into a computer can simplify a lot of data entry, which was handled before by using keyboards. Handwriting recognition seems to be more practical than speech recognition because of the fact that in crowded rooms or public places one might not wish to speak to his or her computer due to the confidentiality or personal nature of the data. Another reason is that it might be annoying to others if someone keeps speaking to his or her machine. It is also already possible to have handwriting recognition in very small hand-held devices, while a speech recognition system is not yet suitable for use as a hand-held machine. However, on the contrary, in term of speed of data entry, speech system is apparently faster and it is much easier to dictate something than to write it.

Pen-based interfaces in digital devices are popular lately and will play a more important role in human computer interfaces in the future. In personal digital assistant (PDA) which is a small handheld device, built-in pen-based handwriting recognition system is already used as an input method. The input method is interfaced to the applications in the PDA, such as personal agenda, address book and communication facilities. In personal computers, pen-based input device (pen or stylus and a pad) is sometimes used to replace the cumbersome mouse for handwriting capability and its small footprint.

Automatic handwriting recognition is the transcription of handwritten data into text in digital format, for use by the computer. The area has been under investigation since the 1950's. Since then there has been steady research effort into the area. Two categorizations are possible; first, in term of processing domains, second, in term of usage categories. Handwriting recognition can be categorized into two domains; online recognition, used in the pen-based interface or offline, used in automated recognition system for processing cheques, forms and the like. Figure 1.1 shows the difference between the two domains. In online handwriting recognition, handwriting signals are captured from the pen traces on the surface of a writing pad. The signals are the input to the recognizer, which then gives out the text of the handwritten input. In off-line handwriting recognition, static images of words written are used instead in the process. A difference between the two is that on-line handwriting recognition requires fast and immediate processing while off-line recognition can be performed within quite a relaxed time constrain. However, recently, this might not

always be the case because it is possible to collect forms containing online handwriting and then to process them in a batch system.

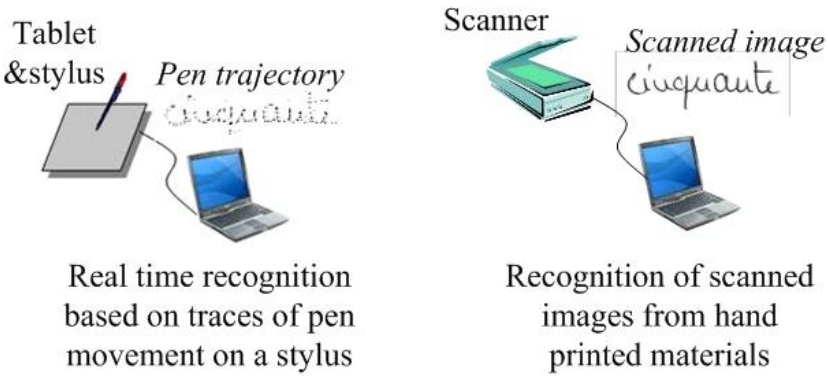


Figure 1.1 Online vs. Offline handwriting system

In term of signals, online signals are normally the pen trajectories, recorded as the x and y coordinates of each point together with eventually the pressure and the time at each point, while offline signals are the image files recorded in a particular image format such as tiff or jpeg. Figure 1.2 below shows the differences between the two signals.

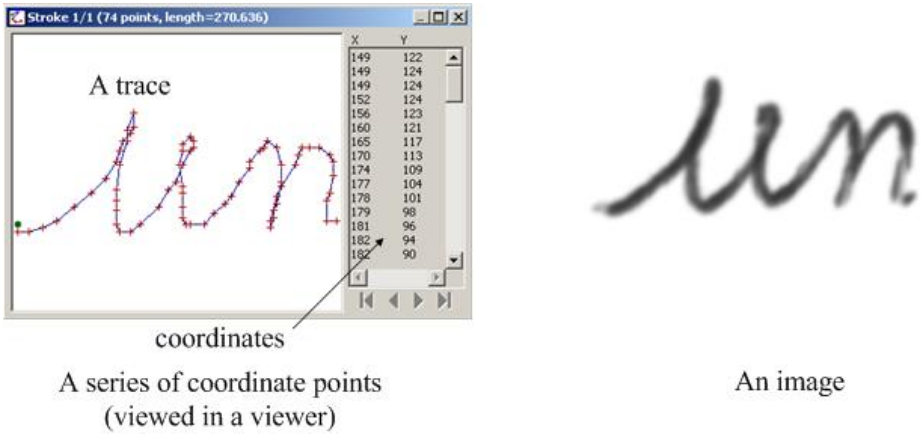


Figure 1.2 Offline signal and Online signal

In the second categorization criteria, (Leedham, 1994) categorizes the automatic processing and recognition of handwriting into the categories as shown in Figure 1.3.

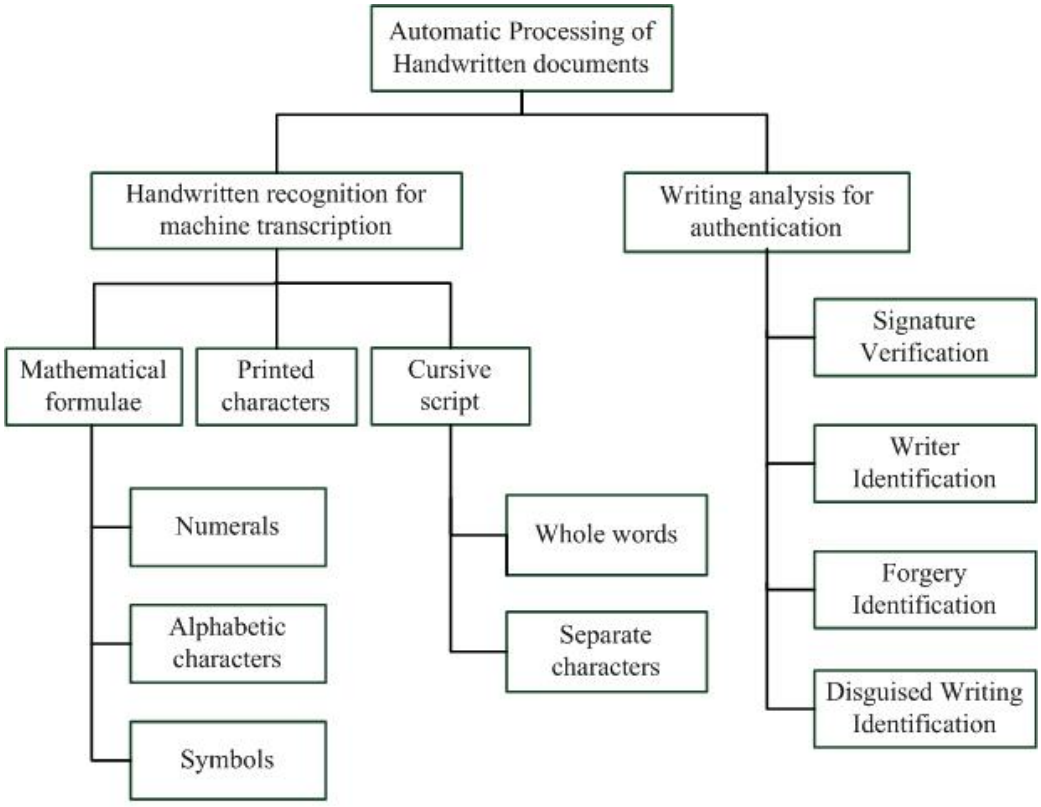


Figure 1.3 Categories of Handwriting Processing

As seen in the diagram, handwriting processing can be divided into two major groups: (a) handwriting recognition and (b) handwriting analysis. Handwriting recognition aims to produce output for machine transcription. It can involve handwritten mathematical formulae, printed characters or cursive handwriting. Handwritten mathematical formulae can consist of numbers and alphabets as well as various mathematical symbols. Printed characters and cursive script handwriting involves whole words or separate characters or combinations of partly cursive and separate characters. Handwriting analysis on the other hand aims at using handwriting for authentication. Among applications in this area are: signature verification, writer identification, forgery identification and disguised writing identification.

Another categorization is given by (Plamondon, 1989) in Figure 1.4, (a more simplified version of Figure 1.3). They divided handwriting into text and signatures. A common application in both text and signature is in using them for verification. In

signature verification, handwritten signature is checked whether it belongs to a particular writer or not and does not normally identify the symbolic classes of characters in the signature. Signature identification is a biometric technique for personal identification where genuine signature signed by an authorized person is compared with the input signature of a person to be identified.

Other pen computing related applications closely related to handwriting recognition is mathematical formula recognition where not just characters are recognized, their layout are also taken into account. Another one is in handwritten document retrieval, but here the so-called ink matching, does not identify the character classes. Finally, handwritten sketch recognition, is based mostly on non character data and typically ignores linguistic information.

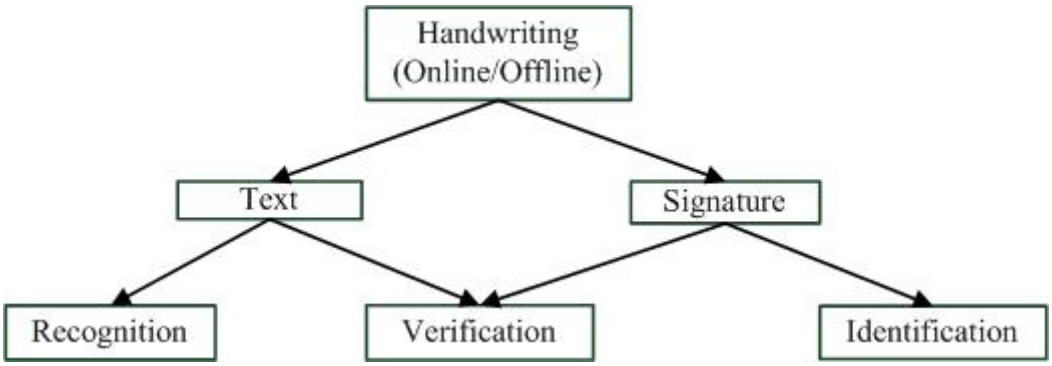


Figure 1.4 Handwriting Recognition (Plamondon, 1989)

1.2 Limitations of Handwriting Recognition System

Although there are many applications of handwriting recognition in both online and offline domain, the technology is not fully matured. There are many improvements that can still be made to make handwriting recognition more widely accepted in computer based applications. In online handwriting, the input signal consists of a time sequence of strokes. A stroke is the writing from the time when

the pen is touched down (pen down) to the time it is lifted (pen up). The characters in the writing signal are usually written in sequence, one character being completed before beginning the next, and the characters typically follow spatial order, from left to right except in certain characters like dots (i's and j's) and crosses (t's and x's). In these cases, the underlying portion of a word is first written, and then the word is completed by writing the crosses and dots. The presence of these delayed strokes posed some problems which if not handled will not provide a good recognition of input handwriting (Figure 1.5).

In some applications and devices, in order to provide good recognition performance, constraints need to be imposed to user input, such as in the way in which handwriting need to be done. An example is in the "Graffiti" system used in the Palm devices (Figure 1.6). Generally, there is no system that can be used in all environments. Each system is somewhat constrained to work in a particular target environment.

Figure 1.5 Problem with handwriting recognition systems

In many researches in handwriting recognition, other than constraining handwriting styles, those that work on unconstrained handwriting input, address only on a few specific areas such as writer dependent systems or systems that utilize only some special small lexicon. For example, they might only cater for the recognition of handwritten characters or numbers and recognition of words from a small specific lexicon. As constraints in the handwriting are reduced, the problem will become more complex because the recognition system needs to handle various limitations, thus, this will affect the recognition accuracy.

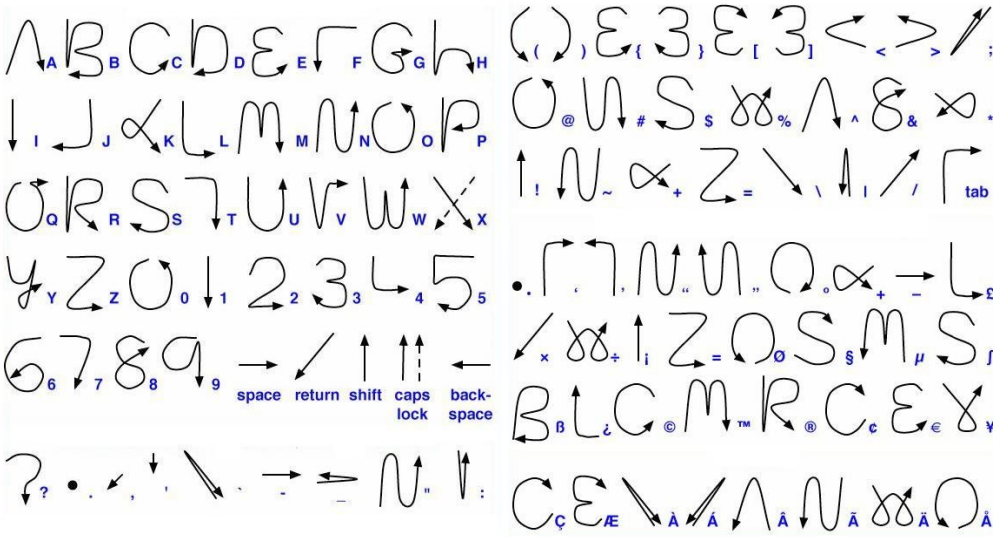


Figure 1.6 Example of constraints imposed in Palm Grafitti handwriting recognition system

1.3 Overview of Handwriting Recognition System

There are many different techniques for handwriting recognition. One generic model of handwritten word recognition system that can be used for our discussion is as follows. The description does not describe a standard but it is typical of most present recognition systems. Figure 1.7 gives a graphical summary of the description.

The input to the system is the word to be recognized which is a word image in the case of off-line and a series of captured information representing the pen trace (the strokes or characters) of the word in the case of an online system. The discussion is similar for both offline system and online system except that the nature of the corresponding implementation of each process might be a little bit different. The output of the system is a text representation of the input word signal presented to the system. In the model, there are three main components; the front-end module, the recognition module and the post processing module. Each module performs their required functions depicted as sub modules within the modules.

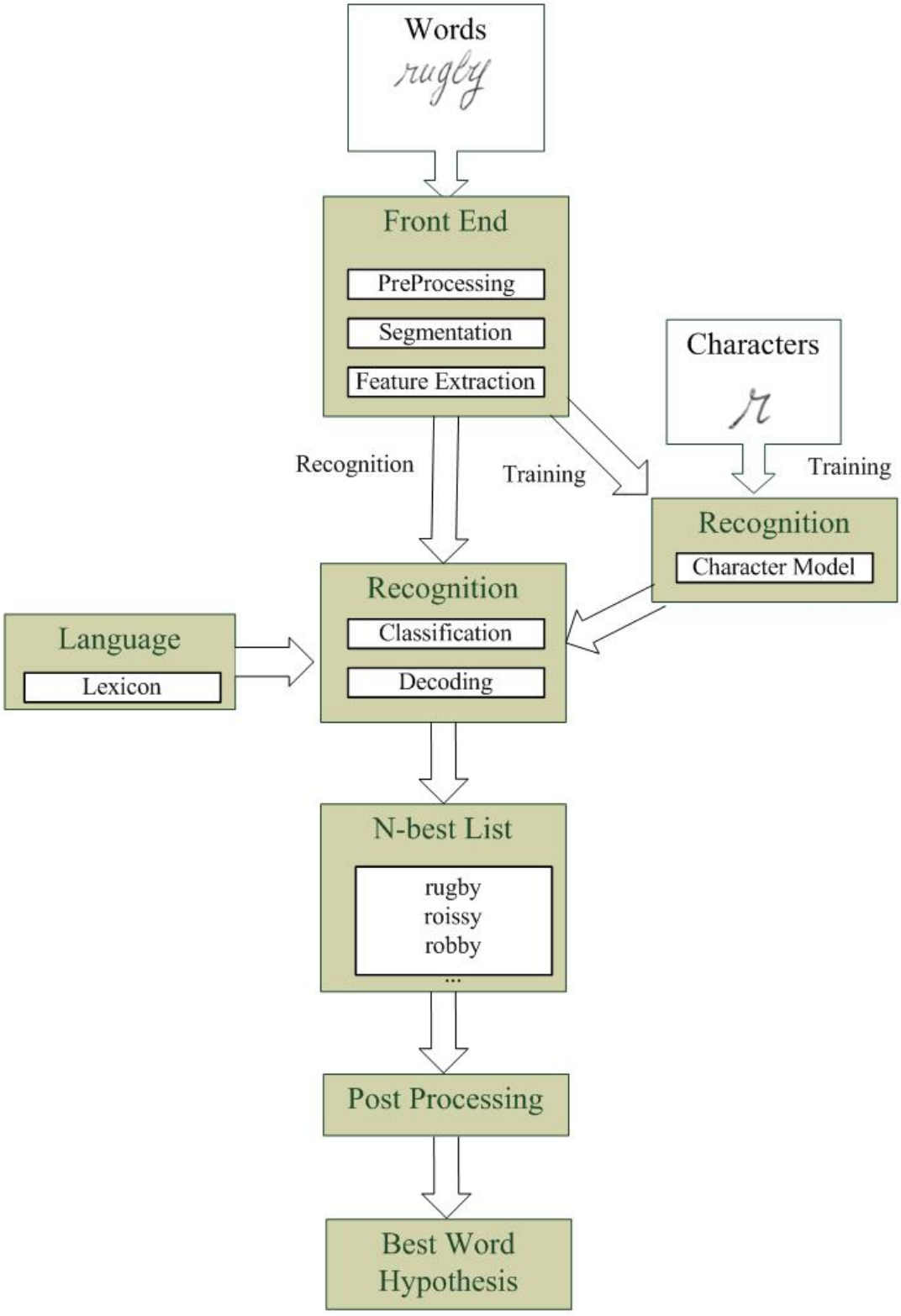


Figure 1.7 Typical Handwriting Recognition System

First, the unknown handwritten word presented to the system needs to be transformed into a form understandable to the recognition computation engine. The front-end module needs to extract information from the presented word in the most efficient form for presentation to and processing by the recognition module. In the front-end processing, the word signal, first needs to be preprocessed to remove undesired variability that will cause difficulties in the recognition process. Operations like reference lines detection and correction to some variations like rotation, size and slant are performed. Secondly, words are segmented into a sequence of basic recognition unit such as characters or parts of characters. Most systems will perform this step; however, there are some that do not. In that case, the word is treated as a whole and recognition is a global process where characters are not first recognized. Thirdly, the segmented preprocessed unit needs to be transformed into a compact feature representation. This process involves extracting discriminant features to build up a list of feature vectors to be used in the recognition stage.

The recognition module in the system involves using a trained module that recognizes basic individual units mentioned earlier and their concatenation in the formation of the word. The word recognition, as will be described in the next section, includes a comparison of the test pattern (the observed word) with each class reference pattern (words in the lexicon) and measuring a similarity score (in the form of distance or probability score) between the test pattern and the similarity pattern. The pattern similarity score is used to decide which pattern best matches the unknown pattern. The implementation of this recognition module in previous systems have been in a number of ways such as dynamic programming, hidden markov model, neural network, expert system, k-nearest neighbor and other combination of techniques. Normally, the process of recognition provides a list of N-best word hypotheses where N can lie between 1 and 10. The list can be further post processed to obtain a better list of word hypotheses. This approach taken during the stage of recognition falls under the category of statistical pattern recognition, the basis of which will be described in the following section.

The post-processing module is used to verify the N-best list and may also perform rejection of unlikely hypotheses. With the help of some source of

knowledge in the form of a language model, some improvements in recognition can be obtained. A language model can be the lexicon, which is a library or list of possible words for recognition, or the words that are allowed as input to the recognition system, but can also include some statistical or structural properties of a given language.

1.4 Statistical Pattern recognition

As mentioned in section 1.3, at the recognition stage, the problem of handwriting is largely statistical in nature. It can be described by the following, according to statistical pattern recognition concept.

The goal of word recognition is to find the most likely word representing the given handwriting signal or image. If O is the observation sequence of a word signal, and W is the word in the lexicon, then the recognition system must choose a word $\hat{W}$ that maximizes the probability that the word W was written given that the observation sequence O was observed:

$$\hat{W} = \arg\max_{W} P(W \mid O) \qquad \text{(Eq. 1.1)}$$

.

$P(W \mid O)$ is called the posterior probability. It is difficult to compute the above maximization. However, it can be simplified by using Bayes theorem, which states that:

$$P(W \mid O) = \frac{P(O \mid W)P(W)}{P(O)} \qquad \text{(Eq. 1.2)}$$

where the probability $P(O \mid W)$ is the probability that the observation sequence O was observed if a word W was written or the likelihood of the observation. $P(W)$ is the a priori chance of the word being written and *P(O)* is the evidence or

normalization factor which represents the unconditional probability of the input signal. *P(O)* can be defined as follow:

$$P(O) = \sum_{k=1}^{K} P(O \mid W_k) P(W_k) \qquad \text{(Eq. 1.3)}$$

where W_k, $k=1,2,...,K$ are the words in the lexicon and K is the total number of words. It is a scale factor that ensures that the posterior probabilities sum to unity. However, $P(O)$ is normally omitted because this term is common across all words.

Therefore, ignoring $P(O)$, for a given new word signal input O, classification is made by selecting the word corresponding to the largest value of $P(W \mid O)$, that is:

$$\hat{W} = \arg\max_{W} P(W \mid O) = \arg\max_{W} P(O \mid W) P(W) \qquad \text{(Eq. 1.4)}$$

Thus, the probability of the word being written is a product of the two probabilities. In short, this can be viewed as the discriminant function for a word, which is formed by joining the likelihood function and the prior probability. The process of finding these two probabilities and then, finding the resulting combined probabilities for all words in the lexicon and selecting the highest probability is what constitutes the recognition system. For a given input O, the probability of misclassification is minimized by choosing a word having the largest value of the product of the two probabilities.

1.5 Problems in Handwriting Recognition

Many researchers have conducted research in handwriting recognition in the last years. Although many problems have been solved, there are still many problems at hand. Despite the availability of computing power and progress made so far, the capability of handwriting recognition system is still incomparable to human recognition. As mentioned earlier, no two humans have exactly the same handwriting and even no two sets of handwritings of the same person for the same

word are exactly the same. Between people, the variability can include the slant, the size of characters, the shape and how cursive or disjoint the characters in the handwriting are. Variations in handwriting can also be in term of the applications, even if for off-line handwriting applications such as form processing, handwritings are normally guided by boxes. Figure 1.8 shows a random sample of handwriting taken from IRONOFF database that demonstrates these differences. Figure 1.9 adapted from (Tappert, 1994) show further variations.

As mentioned in section 1.2, handwriting recognition can be performed by taking a word itself as a whole entity for recognition. This method has been used by a number of researchers. The model for recognition is the whole word model, which are trained to cater for variations and similarities within word such as co-articulation. Because the whole word is taken in training the system, segmentation is avoided. However, word model recognition is only applicable in cases where the lexicon is small.

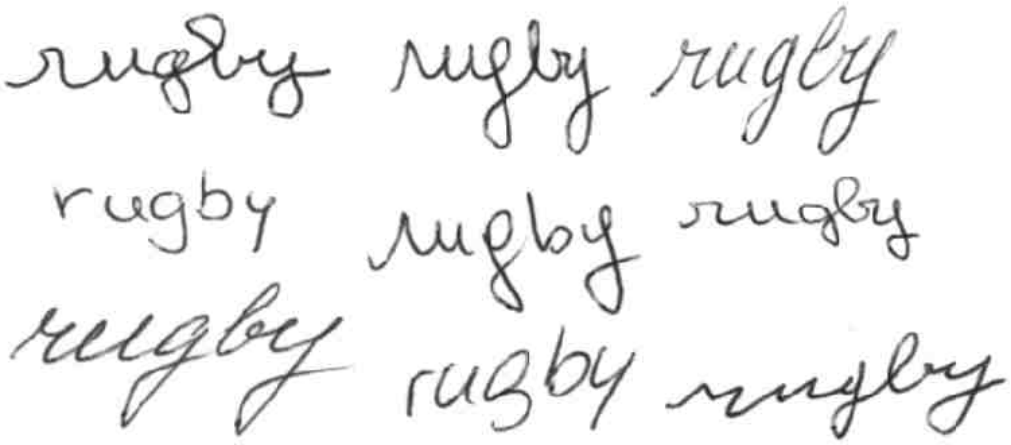


Figure 1.8 Variations in handwriting style –random sample of handwriting taken from IRONOFF database

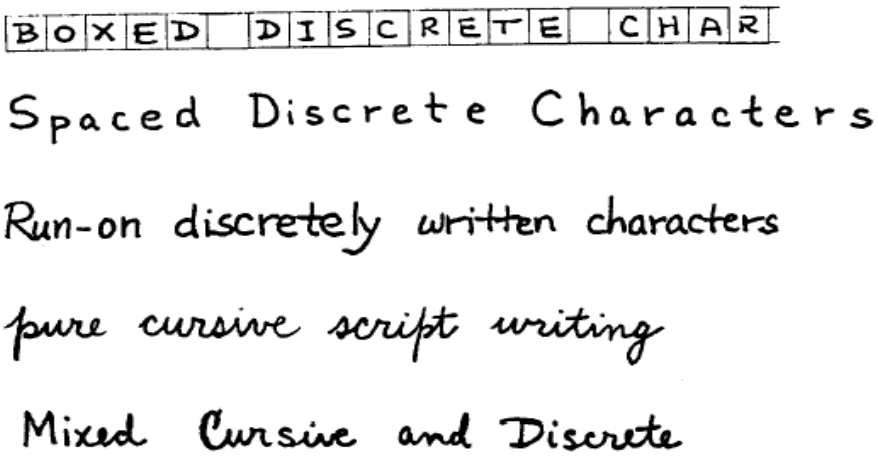


Figure 1.9 Types of handwriting. Adapted from (Tappert, 1994)

For larger sized lexicon, whole word recognition is unfeasible. This is because since each words are treated individually, the recognition information they contain cannot be shared among words, which means that the larger the size of the lexicons, the larger the recognition model is. Because of this limitation, many word recognition systems uses smaller units as the basic model, either a character or an entity smaller than a character (sometimes termed as pseudo character or grapheme). Model used for word recognition is the concatenation of the basic model at the character or sub-character level. In this way also, having larger lexicon does not mean a bigger number of word model. It simply means constructing the new word model from existing smaller sub-unit model.

Segmentation is then required to cut the words into smaller units. This creates another set of problems. Some words are written very cursively but some writers prefer to write disjoint characters when writing a word. Yet some, mix between cursive writing and disjoint without any particular order or rule, as can be seen in figure 1.8 and 1.9. In segmentation based word recognition system, the lexicon plays a very important role in the segmentation process itself as the recognition process determines the best and definitive segmentation points.

1.6 Recognition Modeling.

There are many different methods of training and modeling of a handwriting recognition system. A few of them of relevance to the thesis will be discussed here with the aim of comparing the pros and cons between them and eventually focusing on the method that have been used in this thesis. Among the existing methods are Hidden Markov Model (HMM), Neural Network (NN), Expert System, k-nearest neighbor and other techniques or combination of techniques. Some researchers divide these methods into two main descriptions; the syntactic, which involves describing character shapes in an abstract fashion and the statistics methods where the system learn from data directly without the implementer having to specify explicitly the structure or the knowledge into the system. HMM and NN falls under the statistical method. Support Vector Machine (SVM) which will be the focus of this thesis is another.

1.6.1 Hidden Markov Model

In many handwriting recognition systems, the basic modeling component for recognition is the Hidden Markov Model (HMM). This follows from its success in speech recognition. The ability to statistically model the variability of handwriting is its major strength. HMM uses Markov process, represented as a state machine to model the temporal evolution of handwriting. The probability distribution associated with each state in an HMM, models the variability in the handwriting.

In this section, a very brief description of HMM is given, to facilitate explanations in this introductory chapter. Detail accounts will be given later in chapter 3. Figure 1.10 depicts a simple five state HMM. It can be attributed with the following parameters: N – the number of states in the model (5 states), probabilities of transition between states denoted by matrix A (consisting of probabilities a_{11}, a_{12}, etc) and probability of emission or output denoted by matrix B (consisting of probabilities $b_1(o_t)$, $b_2(o_t)$, etc where o is the input observation). B represents the probability of observing an input feature vectors in a given state.

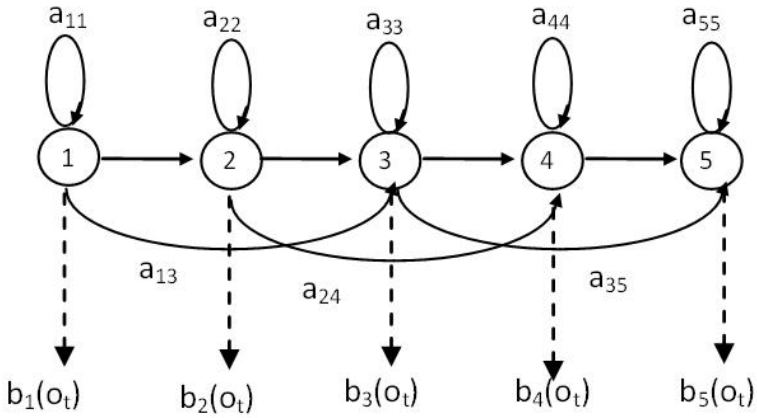


Figure 1.10 Example of a 5 state HMM

HMM uses Bayes classifiers, which gives likelihood ratio for classification. The output probability distributions could be parameterized in either discrete or continuous distribution. The choice depends on the level of modeling accuracy needed and the amount of training data available. Continuous distribution is more accurate, with the disadvantage of complexity in modeling and computation. The most commonly used probability distribution function used in HMM is the multivariate Gaussian distribution as follows:

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n \left|\sum_j\right|}} \exp\left(-\frac{1}{2}(o_t - \mu_j)' \sum_j^{-1} (o_t - \mu_j)\right) \qquad \text{(Eq. 1.5)}$$

Where n is the dimension of the observation vector o_t at time t and the subscript j indicates that the Gaussian under consideration belongs to the j^{th} state of the HMM. $\sum_j$ is the covariance matrix which normally is taken to be a diagonal matrix taking assumption that each feature components are independent of each other.

Using HMM, training is thus, estimating the parameters of the recognition model, using some parameter estimation model. One popular method is by using maximum likelihood estimation (MLE) to maximize the probability distribution that models the observation vector in the modified form of equation 1.5. The

Expectation maximization (EM) algorithm is normally used in the MLE procedure. MLE procedure however, lacks discriminative power because only in-class data is used in modeling a particular class model. Fitting Gaussian and applying Bayes rule for classification also does not give an optimal classifier. There is always some classification error because the decision threshold always occurs inside one of the classes, which could be due to some overlapping features in the feature space of the classes. To obtain discriminative power, HMM could be trained by including out of class data, using a number of optimization methods such as minimum mutual information (MMI) estimation or the minimum classification error (MCE); two issues that will be discussed in chapter 3 .

1.6.2 Neural Network

Neural network (NN) is a discriminative classifier in that all in class and out of class data are used in the training. NN body of literature is enormous because it has been used very widely in many areas. NN have been used in handwriting recognition system with success. However, compared to HMM, they require more computation. In addition, and of more importance, is that, they are not able to model time variations in handwriting signal, which is important for word recognition. They are static classifiers which require fixed size feature vector. Due to that, they are normally used only in character or digit recognition. There are also some other weaknesses of NN as a discriminative classifier, some of which are as follows.

a) In term of generalization property, NN is known to over fit data unless specific measures are taken to avoid that (Ganapathiraju, 2004). Although cross-validation can be done to avoid that, it is quite hard to achieve good generalization when only a limited amount of training data is available.

b) The optimization process in gradient-based NN learning is according to the principle of empirical risk minimization (ERM) using the back-propagation (BP) algorithm (Rumelhart, 1986). Though this guarantees good performance on the training data, performance on the test data is difficult to obtain.

c) Choosing Model Topology is another issue. In most connectionist system, the topology is needs to be fixed prior to training (Bodenhausen, 1993). Often, this requires some expert knowledge of the data. Learning the topology automatically is possible but quite time and resource consuming.

d) Training Convergence is considerably slower (as compared to ML estimation in HMM). In fact, NN training (for that matter, MLE training also) does not guarantee global optimum.

1.6.3 Syntactic Modeling technique

In Syntactic Modeling technique, handwriting recognition is based upon the idea that character shape can be described in an abstract fashion. Using expert system is one of the methods. Generally, an expert system incorporates human knowledge about the problem domain into stored knowledge. It basically consists of knowledge acquisition part that obtains knowledge and expertise from human experts in the form of rules, the knowledge representation part that provides methods used to represent human knowledge and expertise in the computer system and knowledge inferencing part that applies stored expertise to make decisions.

Syntactical handwriting recognition does not require a large amount of data for training, not as much as used in statistical handwriting recognition. The success of this method has largely been limited because of the complexity and ambiguity of handwriting styles and difficulty in formulating general and reliable rules as well as in automating the generation of these rules from a large database of characters and words. However, this approach has been revived recently with the use of fuzzy rules and grammars that use statistical information on the frequency of occurrence of particular features (Parizeau, 1995) (Malaviya, 1994) (Anquetil, 1997).

1.6.4 Support Vector Machine

A good classifier needs to have good generalization, minimum risk, better convergence properties and better discrimination power and possess a model

topology that does not have to be fixed a priori. This has led to the support vector machines (SVM) which is the focus of this thesis.

SVM had been proven to generalize well. SVM generalization properties allows for a bound on performance on a given test set to be part of the training process without having to actually test the system. Normally, empirical risk minimization (ERM) as used in NN is the most common optimization criteria used to estimate classifiers. However, using ERM, the solution is not unique. There are several configurations of the classifier that can achieve minimum risk specified in the ERM, on the training set (as seen in NN training). There is a need to decide on the configuration that has the least upper bound on the expected test set error. This is the principle of structural risk minimization (SRM). Support vector machines are based on this principle. With SRM, a classifier will have the least expected risk on the test set and therefore a good generalization.

This section introduces SVM but in chapter 4, the theoretical principles of SVM will be discussed in detail. In the simplest form, SVM is a linear binary (2-class) hyper plane classifier. For a non-linear case, SVM implicitly transform the non-linear data to a high dimensional linear space and construct a linear binary classifier in this space. This is done implicitly, without having to perform any computations in the high dimensional space. Because of this, data of high dimension or even sparse data pose no problem when implementing SVM. The eventual hyper plane in the high-dimensional transformed space actually results in complex decision surfaces in the input data space.

There have been many successful cases of using SVM in many problems, classical or new. In most cases, SVM consistently performed better than other non-linear classifiers. Initial use of SVM reported was in classification of handwritten digit. However, widespread usage of SVM was initially hampered by the unavailability of an efficient optimization method, which can handle large data efficiently and fast without consuming much of computer memory. With ongoing development of efficient optimization methods, SVMs now handle the problem. There are many applications of SVM up until now, in many areas, too many to list all. Randomly picked list follows: e-learning, text classification, handwritten

character recognition, handwritten character categorization, image clustering, speech recognition, speaker verification, forecasting, fraud prediction, protein structure prediction, land cover classification, intrusion detection, cancer prognosis, particle and quark-flavor identification in high energy physics, object detection, text categorization and time series prediction. A detail reference will be provided in chapter 4.

In speech recognition, SVM was used with HMM in the first SVM-based large vocabulary speech recognition system (Ganapathiraju, 2002). The hybrid system uses HMM to handle the temporal evolution of speech and SVM to discriminatively classify frames of speech. It was a first successful application of SVMs to continuous speech recognition. The system improves performance over traditional HMM-based systems. The hybrid system achieves a 10% improvement relative to an HMM system, which is significant.

Recent application of SVM in handwriting recognition was mainly at the character recognition level. Usually, SVM (with kernel) are designed to deal with data of fixed dimension. However, on-line handwriting data is not of a fixed dimension, but of a variable-length sequential form. In this respect, SVMs cannot be applied to HWR in a straightforward manner. (Bahlmann, 2002) uses a special SVM kernel for sequential data, the Gaussian dynamic time warping (GDTW) kernel that instead of the squared Euclidean distance in the usual Gaussian (RBF) kernel, it uses the dynamic time warping distance. Bahlmann achieved superior recognition rate in comparison to an HMM-based classifier.

The last two applications of SVM described earlier provide the motivation for us to research into using SVM for handwriting recognition at a higher level. The author has already used SVM for character recognition and have achieved more satisfactory result than Bahlmann in term of character recognition using SVM (Ahmad, 2004b). A further investigation into using SVM in a hybrid system of SVM and HMM similar to the work of Ganapathyraju is the objective of this thesis.

1.7 Scope and Objectives

The author has presented a brief description of the background of the issues and problems in handwriting recognition. A brief description of the tools used in handwriting recognition has also been given. Many problems are still not satisfactorily solved. Due to the wideness of the scope that falls in the arena of handwriting recognition, to tackle them all will require immense resources. Thus in this thesis, the focus is on a specific issue relating to improving the handwriting recognition system using new methods. The author has focused on the recognition aspect in a handwritten word recognition system. The aim is to investigate whether it is possible to increase the word recognition accuracy using segmentation based recognition method in the context of a hybrid system. Due to the emerging use of the learning method of Support Vector Machine (SVM) and the immense popularity of Hidden Markov Model (HMM), the author has chosen to investigate the effectiveness of using SVM in character recognition itself and its use in a hybrid environment of a segmentation based handwritten word recognition system. In this system, the discriminative property of SVM is exploited in tandem with the class representative property of a HMM.

The primary goal of this thesis is to propose a hybrid SVM/HMM handwritten word recognition system that caters for a medium sized lexicon. The system should be able to handle connected cursive handwritten words. The system borrows some ideas on existing systems based on discrete HMM and a hybrid of Neural network and HMM. Although the eventual aim is to adapt a simplified system based on word level discriminant training, in this thesis, emphasis is put in character level discriminant training, due to the difficulty in deriving correcting gradient from word level to character or sub-character level training. The final product is a working system which proven the concept and the tests done gives some ideas as to what are the problems and the recommendations in developing such a system.

The main contributions of this thesis are as follows:

a) Formulation and parameterisazion of SVM for handwriting recognition problem.

b) Testing of SVM on major character database, proving the effect of various parameterizations in improving character recognition.

c) Adaptation of SVM for posterior probabilistic measures output.

d) Method for segmentation and feature extraction of character segments from online word signal.

e) Use of SVM in a hybrid situation with HMM in improving the discrimination ability of the overall recognizer.

f) Comparison of SVM/HMM hybrid implementation with other hybrid systems in handwriting recognition.

1.7.1 Thesis Layout

The main content of this thesis is divided into 7 chapters. This first chapter presents some background, the issues related to handwriting recognition and the scope, aim and contribution of this thesis. Chapter 2 presents the state of the art of handwriting recognition. Pattern recognition concepts and statistical pattern recognition issues are first introduced. Then speech recognition issues which had direct influence in handwriting recognition is discussed, followed by online and offline handwriting recognition issues. Online handwriting recognition using Support Vector Machine and Hidden Markov Model are then elaborated. Further issues on the use of SVM for character recognition are then discussed.

In Chapter 3, theoretical foundation of Hidden Markov Model (HMM) is discussed. After introducing Markov chain, HMM parameter estimation and training are presented. Forward-backward, Baum-Welch, Viterbi algorithms and the usage of HMM in discrete handwriting recognition system are discussed. The uses of HMM in hybrid handwriting recognition systems are then given some accounts.

SVM is discussed in Chapter 4. The theoretical foundation of SVM is presented here. Issues like Empirical Risk Minimization (ERM), Structural Risk Minimization (SRM) and the concept of maximal margin classifier are introduced. Aspects of SVM estimation and training are then discussed. For the adaptation of SVM in the hybrid system, SVM probabilistic output is discussed. Then, the use of SVM for handwritten character and word recognition is presented.

Chapter 5 provides an overall description of the online handwriting recognition system. After an overview, preprocessing is presented, followed with the training and recognition procedure in the SVM/HMM hybrid system. Chapter 6 outlines the experiments conducted and the experimental results. After providing some details about the overall databases in handwriting recognition and the ones used (UCI, MNIST, IRONOFF and UNIPEN), a description about the experiments conducted is given. Results for SVM Selection, the use of SVM in character recognition and the use of SVM in two other related areas that the author is involved in, namely mathematical symbol recognition and fraud prediction are discussed.

Word recognition results using the hybrid SVM/HMM system are then given in chapter 6 together with some ad hoc comparisons with the results of other comparable systems, namely the ANN/HMM offline system and the hybrid SVM/HMM used in speech recognition. Error analyses of the system are also given in this chapter. Finally, in chapter 7, a conclusion and suggestion for future work are given. Figure 1.11 summarizes the thesis layout in graphical form.

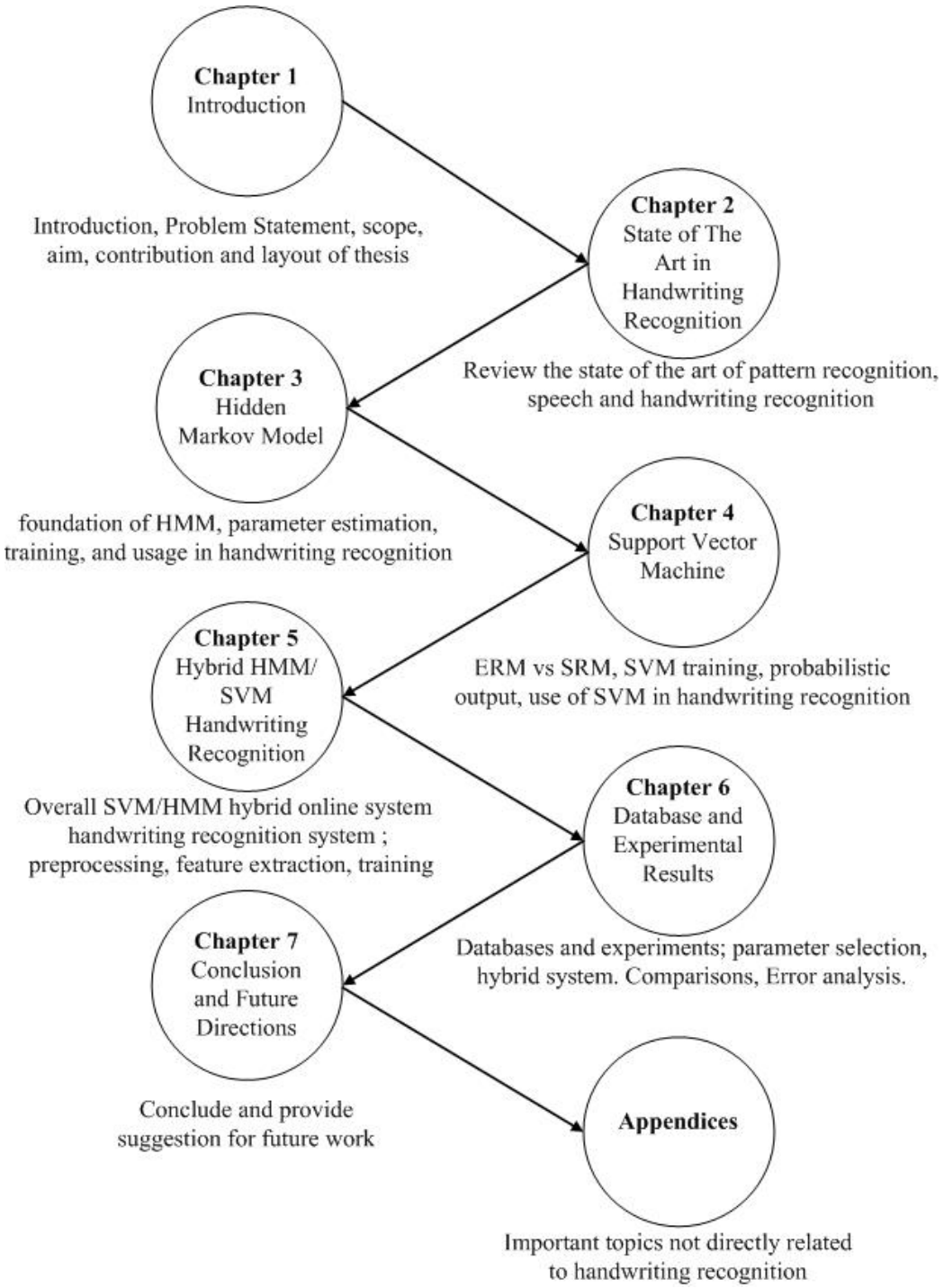


Figure 1.11 Thesis Layout

CHAPTER 2

STATE OF THE ART IN HANDWRITING RECOGNITION

2.1 Introduction

In this chapter, a review the developments in the field of handwriting recognition are made. First, an introduction to pattern recognition concepts and statistical pattern recognition issues is given. As developments in speech recognition have had direct influence on handwriting recognition, the author review some aspects of the developments in speech recognition research in relation to handwriting recognition research. Then a look into the developments in handwriting recognition is made, both online and offline including elaborating on a few issues addressed earlier in the first chapter. In both domains, much effort has been spent on the development of classification methods and algorithms. The most notable aim is to increase recognition accuracy or reduce error rate, while taking into consideration memory requirements and computation complexity. At the end of this chapter, a section is devoted towards discussing the perspective of the use of Support Vector Machine (SVM) in speech and handwriting recognition.

2.2 Pattern Recognition

Handwriting recognition is an application in the field of pattern recognition. Automatic recognition, description, classification and grouping of patterns are important problems in many engineering and scientific disciplines. (Watanabe,

1985) defines pattern as the opposite of chaos; an entity, vaguely defined that could be given a name, for example a fingerprint image, handwritten cursive word or human speech. Pattern recognition (PR) is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns. The primary goal of pattern recognition depends on whether it is a supervised classification or unsupervised classification. In *supervised classification*, the input pattern is identified as a member of a predefined class. In *unsupervised classification* the pattern is assigned to an unknown constructed class (Jain, 2000). Handwriting recognition really falls under supervised classifications as handwriting examples are used in building the recognizer.

A model for pattern recognition is shown in Figure 2.1. As can be seen it is operated in two modes: training (learning) and classification (testing). A PR system needs to be trained to obtain a recognition or classification model for use during classification. In training mode, features representing input patterns are extracted and used for training to partition the feature space. Some feedback from the learning stage allows the optimization of the preprocessing and feature extraction or selection strategies involved. At the end of training, parameter values for the classifier are obtained. In classification mode, the trained classifier is used to classify the input pattern into one of the pattern classes.

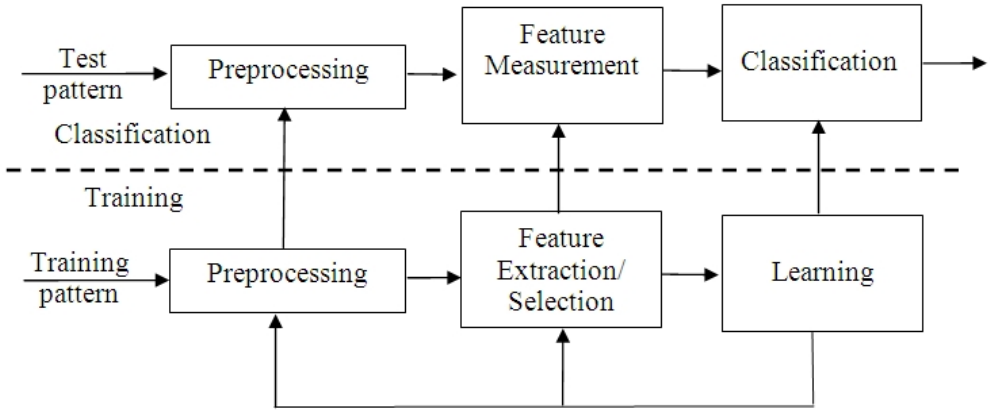


Figure 2.1 A model of Pattern Recognition System

Depending on complexity, some PR application might require extensive computation during training, especially in applications involving the processing of large data. There can also be huge data sets needed during the training stage of the systems. There are many approaches in PR but it is important to note that there is no single optimal approach for all and multiple methods and approaches might need to be combined and used in a single system. This applies to handwriting recognition as well. Figure 1.7, presented in chapter 1; match this model of pattern recognition system.

2.2.1 Learning Approaches in Pattern Recognition Systems

PR systems can use various approaches to learning. (Jain, 2000) summarizes four general approaches of pattern recognition. They are template matching, statistical classification approach, syntactic or structural matching and neural networks (Jain, 2000). Such approaches are neither necessarily independent nor disjointed from each other. Occasionally, a technique in one approach can also be considered to be a member of other approaches (Bortolozzi, 2005).

In template matching, a simple generic operation is used to determine the similarity between two entities of the same type (such as groups of pixels, shapes, curvatures, etc). A template or prototype of the pattern to be recognized is matched against the stored template. Matching techniques can be grouped into three classes: direct matching (Gader, 1991), deformable templates and elastic matching (Dimauro, 1997) and relaxation matching (Xie, 1988).

In Statistical Approach, the concern is with statistical decision functions and a set of optimal criteria, which determine the probability of the observed pattern belonging to a certain class. Many handwriting recognition approaches belong to this domain, such as : k-Nearest-Neighbor (k-NN) classifier (Mico, 1999), Bayesian classifier (Duda, 2001), Polynomial Discriminant classifier (Schurmann, 1996), Hidden Markov Model (HMM) (Rabiner, 1986b), Fuzzy set reasoning (Gader, 1996) and Support Vector Machine (SVM) (Burges, 1998).

In Syntactic approach, a pattern is viewed hierarchically. A complex pattern can be composed of simpler sub patterns, which are in turn built from yet simpler sub patterns (Fu, 1982). The simplest sub pattern is the primitive. Structure of a pattern can be compared with the syntax of a language where a pattern is viewed as a sentence of a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a number of primitives and grammatical rules can be used to describe a collection of complex patterns where the grammar for each pattern class is inferred from training samples. In relation to handwriting recognition, structural methods can be categorized into two classes (Bortolozzi, 2005): grammatical methods (Shridhar, 1986) and graphical methods (Kim, 1998).

Finally, a Neural network (NN) can be viewed as a massively parallel computing systems with large number of processors and interconnections. Zhang (Zhang, 2000) gives a comprehensive review. NN models learn complex nonlinear input-output relationships using sequential training procedures in a network of weighted directed graphs of nodes and directed edges (with weights). The main advantages of neural networks is that it can be trained automatically using examples, gives good performance even with noisy data and can be implemented in parallel. The most widely studied and used neural network is the Multi-Layer Perceptron (MLP) (Bishop, 1996). The most popular neural network classifier and most frequently used traditional classifiers is the MLP trained with back-propagation (LeCun, 1998b). Other types of NN include Convolutional Network (CN) (LeCun, 1998a), Self-Organized Maps (SOM) (Zhang, 1999), Radial Basis Function (RBF) (Bishop, 1996), Space Displacement Neural Network (SDNN) (Matan, 1992b) and Time Delay Neural Network (TDNN) (Lethelier, 1995).

2.3 Developments in Speech Recognition

Speech recognition, also known as automatic speech recognition (ASR) converts spoken words to machine-readable input (transcript an acoustic speech signal into its equivalent textual form). It is used to interact with a computer, similar to textual input through a keyboard. It was supposed to replace, or reduce the reliability on, standard keyboard and mouse input. With that it should assist people who have little

keyboard skills or dyslexic people or people with physical disabilities that affect either their data entry, or ability to read or check what they have entered.

In ASR applications such as phone-based automated timetable information, or ticketing purchasing, the user makes contact with the system, and speaks in response to commands and questions. Most ASR breaks down the spoken words into phonemes and analyzes them to see which string of these units' best fits an acceptable phoneme string or structure that the system can derive from its dictionary.

Speech recognition technology is more mature compared to handwriting recognition technology. The technology of ASR and transcription has progressed greatly over the past decades. Research in ASR began in 1936, but it was not commercialized until the early 1980's when Hidden Markov Model (HMM) technology was introduced (Rabiner, 1986b). HMM has been the dominant approach to speech recognition since then. However in the late 1980's, there seems to be a shift towards Neural Network, in particular Multilayer Perceptron (MLP) and Time Delay Neural Network (Weibel, 1989) as well as related method such as Learning Vector Quantization (LVQ) (Kohonen, 1988). This is due to their discriminative ability as compared to HMM which is trained with Maximum Likelihood Estimation (MLE) criteria. Anyhow, the introduction of an alternative optimization criterion such as the Maximum Mutual Information (MMI) (Bahl, 1992) and Minimum Classification Error (MCE) (Juang, 1992) in HMM improved the recognition accuracies in some systems which makes HMM still popular.

Today, speech recognition system which is based on single stand alone NN technology or HMM by itself is not common. Since early 1990's, hybrid systems combining HMM and NN were widely popular. The hybrid system takes advantage of NN for its discrimination ability and HMM for its excellence in sequential modeling (Bengio, 1991) (Rigoll, 1998). A comprehensive survey on this hybrid method can be found in (Bourlard, 1998). HMMs and NNs combined are proven to improve classification capabilities.

2.4 State Of The Art in Handwriting Recognition

Various surveys have dealt with handwriting recognition from many aspects; online (Tappert, 1990), offline (Steinherz, 1999), machine-printed and cursive script handwritten characters (Guyon, 1996) (Plamondon, 2000). Many papers also reviewed or described their specific research in handwriting recognition emphasizing issues within the many components involved in the overall system as described in section 1.3, such as in preprocessing and segmentation, feature extraction, recognition modeling as well as post processing stage.

Research work in handwriting recognition started later than in speech recognition. The advent of the tablets in late 1950's has resulted in active endeavor in handwriting recognition research lasting through the 1960's. However, it ebbed in the 1970's but was renewed in the 1980's. In on-line handwriting recognition, the renewed interest was due to the availability of more accurate electronic tablets, more compact and powerful computers, and better recognition algorithms. In addition, combined tablets and flat screen displays brings input and output together, which further permits the use of electronic ink, that is the instantaneous display of the trace of the motion of the stylus tip directly under the stylus. During this period also, office automation work, coupled with usability and user friendliness has increased interest in more natural methods of entering data into machines. Researcher's then, starts to more clearly understand the applications appropriate for handwriting recognition. In offline handwriting recognition, progress was similar. As the need for automation in the postal and banking industries increased, new developments were made in the field to cater for the needs. This is also aided by progress made in computer processor hardware, speed and memory capacity available in those machines.

During the last twenty years, there were much more development in handwriting recognition research. As mentioned earlier, this in part is very much due to the progress made in the speech recognition methods and algorithms, which have then been adapted into handwriting recognition. Usage of Hidden Markov Model (HMM) has been popular in handwriting recognition in similar context to speech recognition. HMMs gained growing interest in the handwriting recognition research community

because it was already in a mature state in the context of speech recognition. It is straightforward to transfer the HMM approach from speech recognition domain to the handwriting recognition, especially the on-line domain since pen-trajectory data can be viewed as a time series of samples similar to speech signal.

Literatures in handwriting recognition generally divide handwriting recognizers into whole word (segmentation free) or segmentation based. Whole word (or holistic approach) based does not involve segmentation where recognizer look at the whole word, while in segmentation based; words need to be segmented for recognition. Segmentation based word recognizer can either be based on classical analytical segmentation or they are segmented into characters based on the recognition results. In classical analytical segmentation, words are analytically segmented into characters. In segmentation based recognition, words are explicitly or implicitly segmented into characters by over segmenting them into smaller than character slices (or rather a primitive or can be called something else) and later determining the correct characters segmentation alignment using dynamic programming based algorithms. These issues are further discussed in section 2.4.4.

The earlier approach to recognition was based on classical analytical segmentation. In later work, researchers' attempted the segmentation-free with great success; however, they are limited only to a very small lexicon. Then, segmentation approach was cleverly reapplied by taking into consideration that we need to know the word in order to segment it and we need to know the individual characters in the word, in order to recognize it, an idea called Syre's paradox (Steinherz, 1999). That is when segmentation based recognition comes in. There are merits and disadvantages of either explicit segmentation or implicit segmentation which will be discussed later but in both cases, words are not actually presegmented. They are just cut at various places in the word based on certain criteria. The so called segmentation is the results of combining the slices at the right combination and the right points. To find the optimal segmentation points that form the best set of characters, combination of cuts are evaluated by a character classifier to score the combination. Finally, the word with the optimal score is generally found by applying Dynamic Programming or similar techniques.

Speech recognition were initially based purely on HMM with discrete or Continuous Mixture Densities (Rabiner, 1985). Later, hybrid system became popular. Similarly, in handwriting recognition system which is mainly segmentation based, a hybrid NN and HMM became very popular as NN being a discriminative classifier fits well into HMM structure which handles the temporal nature of handwriting to create a better recognition system. The ANN/HMM hybrid were used in recognition at the character level as well as word level. With proper language model, sentence recognition can be handled.

Also coming from successful usage in speech recognition, segmental modeling was later introduced in an attempt to achieve a more realistic modeling of the handwriting signal. In HMM, observation modeling is at the frame level, while in segmental modeling, a segment which is composed of several observations is modeled. In a way a segment corresponds to a homogeneous portion of the signal which typically can be a stroke in a character. Segment modeling allows automatic handling of different handwriting styles (Artieres, 2000). However, segment modeling requires more computation than the classic HMM. They are mainly used in the post processing stage and there are many possibilities of implementing the segment models which makes compromise between flexibility and robustness possible.

Many researches in handwriting recognition systems implementing the methods mentioned are targeted to small–scale and constrained applications where the vocabulary or the lexicon of words are small. For dealing with large vocabulary system, researchers need to handle crucial issues such as improving recognition speed and computational efficiency while maintaining good recognition accuracy. (Koerich, 2002). Speed and recognition accuracy are two aspects of mutual conflict, but have been tackled by using better search strategies, use of verification steps after the coding, better decoding algorithm and post–processing of the N–best candidate list.

In the following subsections, developments in on-line and off-line handwriting recognition are reviewed separately. The author then looks at developments and issues in methods and techniques at the various stages of the recognition system which are applicable to both domains.

2.4.1 Developments in Online Handwriting Recognition

Research in online handwriting recognition started in the 1960s and has been receiving great interest from the 1980s. Tappert e. al. (Tappert., 1988) (Tappert, 1990) reviewed the status of research and applications before 1990, while a recent survey done by Plamondon and Srihari, (Plamondon, 2000) gives an overview of the near recent situation, for both online and offline handwriting recognition, mainly concerning western handwriting. Nakagawa, (Nakagawa, 1990) and Wakahara (Wakahara, 1992) provides some reviews and insights into early works of online Japanese character recognition. Liu et. al. (Liu, 2004) contributed a survey to online Chinese character recognition (OLCCR). Handwriting researchers also tackled on-line handwritings at either character level or higher level at word or sentence level. Methods used for characters can be applied to higher level since they consists of basic low level entity. In this subsection, the author review the developments at all levels.

The world of handwriting is no doubt dominated by English as it is the world major language. Other Western languages that use Latin are as widely used and important too. However, to be fair to the handwriting recognition community, other languages such as Chinese, Tamil, Japanese or Arabic has to also be regarded as important. Latin based handwritings are less complex than those of the other. In the English language, for example, there are only 26 letter alphabets and each letter has two forms, upper and lower case. English has two basic styles of writing which are; printed and cursive script. The average number of letters per word in English language is five. The number of strokes per letter is 2 for upper case, and only one for lower case, even less for cursive handwriting.

In English, the position and size of the letter is important. We can imagine a writing to be written within 4 reference lines; ascender, core, base and descender.

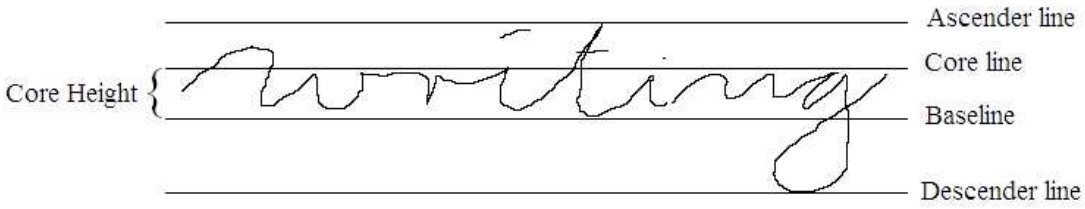


Figure 2.2 English word "writing", written in small letters

Upper case letters sit on the baseline and are full sized. Lower case letters are smaller, and most are about half the height of upper case letters. Some lower case letters have an ascender, which extends upward to almost the height of the upper case letters, some have a descender, which extends down below the baseline, and some have both. These can be seen clearer Figure 2.2 which shows the word "writing" written in small letter and Figure 2.3 for the word "writing" written in capital.

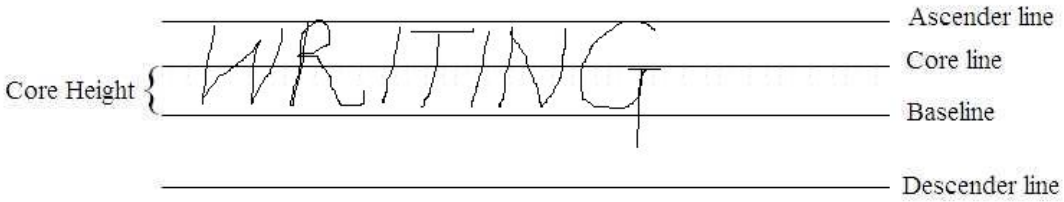


Figure 2.3 English word "WRITING", written in capital letters

Chinese, on the other hand has a much larger character set. A character can also represent a word. There are about 50,000 Chinese characters. They can be written in block or cursive. Chinese characters consist of many strokes because there are a large number of them to be distinguished. The cursive style is written faster and with fewer strokes. The Japanese is in the same category as Chinese since they contain a subset of Chinese characters called the Kanji. Kanji and Chinese characters have essentially the same meaning. Hiragana and Katakana are the Japanese phonetic

alphabet sets with 46 full-size characters each. Kanji, Hiragana and some English alphanumeric makes up a complete Japanese writing system (Tappert, 1990).

Arabic handwriting is used by one-seventh of the world's population, in the languages such as Arabic, Farsi, Urdu, Pashto, Kurdish and Malay. Arabic script consists of 28 basic letters, 12 additional special letters, and 8 diacritics (Biadsy, 2006). Arabic is written from right to left, opposite to Latin based language. Nearly all letters can be written in four different letter shapes depending on their position in a word. Some letters are disconnected and stand alone. Arabic script is similar to Roman in that it uses spaces and punctuation to separate words. It is however different because of the use of dots and strokes which makes the recognition of words in Arabic script more difficult than in Roman script. Many Arabic letters contain dots in addition to the letter body. Strokes can attach to a letter body to create new letters. The dots and strokes are delayed strokes and are written last in a handwritten word. A difference in the dot or stroke can produce a word other than the one that was intended.

The author then review on online handwriting recognition for all languages. However, as there is more research literature for the Western Latin based languages, the bulk of the review will cover on that. A selection of the reviewed literature in online handwriting recognition is given in Table 2.1. In the table, the main authors of the cited literature reviewed and the methods used in the online handwriting recognition system are given. As each system is tailored for certain users and uses a particular database in their testing, the features and the lexicon size in the database used are also given. This can be used to compare the complexity of each system. For cursive handwriting recognition systems, many such systems involve a preprocessor, feature extractor, a trainable classifier and a language modeling post processor.

Early works reported on online handwriting recognition were attributed to (Marmelstein, 1964). Marmelstein's method is based on what is now called the classical analytical method described earlier. It is based on the detection of the down stroke sequences and on a letter-by-letter recognition basis using features such as cusps, closures, and center-line crossings. In recognition, the most likely stroke sequence of the written word is first determined, with reference to the dictionary.

Table 2.1 Summary of Online Handwriting recognition systems

Authors	Method	Features	Lexicon
(Marmelstein, 1964).	Classical Analytical	Stroke sequence	254 words
(Bengio, 1993) (Bengio , 1995)	CNN/ HMM	Annotated image AMAP	25461 words
(Bellagarda, 1995)	HMM/ Kmeans clustering	4 local (slope, curvature) , 3 global	81 character alphabets
(Cho, 1995)	Neuro Fuzzy	direction	letters and digits
(Beigi, 1995)	Discrete HMM /Beam search	5 features - deltax, deltay, tan slope angles, abs. x, abs. y	2000+ words
(Hu, 1996)	HMM	subcharacter stroke (nebulous stroke)	32 English words
(Guyon, 1996)	TDNN/HMM	Slope, curvature speed	25,000 Lexicon
(Artieres, 2000)	Segmental Model	15 features: - 6 temp., 9 spatial	UNIPEN chars.
(Biem, 2001)	HMM /MCE	9 features; local position and curvature info	92 character set
(Artieres, 2000) (Artieres, 2002)	Segment level HMM/trajectory model	36 fixed elementary stroke level rep.	UNIPEN chars.
(Bahlmann, 2002)	SVM with GDTW kernel	3 features per point	UNIPEN chars..
(Bahlmann, 2004)	CSDTW	3 features per point	UNIPEN chars.
(Oudot, 2003)	Activation-verification cognitive model	Geometrical and morphological informations	200 000 words
(Biadsy, 2006)	HMM	3 features per point	Arabic
(Caillault, 2006)	TDNN/HMM	7 features per point	IRONOFF 197 words

Note : All lexicons are English except otherwise stated.

If no match is found, the stroke sequence is modified by accepting less likely stroke sequences until a match is found or until some likelihood threshold value is crossed and recognition attempts stops. Marmelstein's work is important because it emphasizes that more work is required to achieve practical unconstrained script recognition, either online or offline. During that period, not only that recognition is not reliable, but also the speed and cost of the recognition equipment are prohibitive. Due to the complexity of handwriting recognition tasks and lack of computing resources for the computation required, there was not much progress in the research within this area within the next few years running into the seventies.

Based on earlier work (Bengio, 1991) in NN/HMM global optimization in speech recognition (Bengio, 1993), (Bengio, 1995b) proposed a similar system for handwritten word recognition. He uses a combination of convolutional neural network (CNN) and HMM in a global optimization procedure for training the word recognizer. He also proposed word normalization using EM algorithm. For the features, he uses annotated images (AMAP) from the normalized pen trajectory. The replicated CNN spots and recognize characters while the HMM interprets the NN outputs into word score, taking word-level constraints into account. The NN and HMM are jointly trained to minimize an error measure at the word level. The system was called LeRec. Again, based on speech recognition work, (Bellegarda, 1994) described an unconstrained word recognizer using K-means clustering and HMM. He uses 7 features at the character level which consists of 4 local features (tracking slope and curvature) and 3 global features like point distance to base line and distance between penup and pendown whenever they occur.

(Cho, 1995) uses neuro-fuzzy method in his online characters recognition. The idea is to train a number of NN classifiers and aggregating them with fuzzy logic. The method combines the outputs of separate NN with importance of each network, which is subjectively assigned as the nature of fuzzy logic. (Beigi, 1995) developed an HMM-based system for writer independent handwriting recognition using 3 state HMM and beam search. It caters for large lexicon size of more than 20,000 words. (Guyon, 1996) discusses a cursive script recognition system, described within the framework of Weighted Finite State Transductions previously used in speech recognition. It is also a writer independent system that can handle both cursive script and handprint. Time Delay Neural Network (TDNN) is used to estimate probabilities for characters in a word and HMM segments the word in a way which optimizes the global word scores for the given lexicon.

(Artieres, 2000) uses segment models (SM) that model signals at a segment level rather than at observation level. A segment corresponds to portion of the signal which is homogeneous in some sense. For example, in a character, a segment could be a stroke. An observation sequence is assumed to be generated by a succession of SM states, each being responsible for a subsequence. This allows handling of different handwriting styles. (Artieres, 2002) also proposed another flexible

handwriting recognition system that is able to learn easily new symbols and to adapt easily to a specific user handwriting using stroke level HMMs. Each letter is modeled as a stochastic automaton, defined over a set of reference stroke level representations (SLR). This model can easily take into account new letters or writing styles. In all cases, the signal to stroke decoding step remains unchanged, and only the stroke-level system parameters have to be modified.

(Bahlmann, 2002) describes an approach for online handwriting recognition which combines dynamic time warping (DTW) and support vector machines (SVMs) with a kernel he called Gaussian DTW (GDTW). The approach differs with HMM in that it does not assume independence between observations as in HMM and it directly addresses the problem of discrimination by creating class boundaries. Incorporating DTW in the kernel, variable-sized sequential data can be handled by the SVM. (Bahlmann, 2004) again uses SVM in a writer-independent online handwriting recognition system called “Frog On Hand”. The classification/training approach is using cluster generative statistical dynamic time warping (CSDTW). CSDTW is a general, scalable, HMM-based method that holistically combines cluster analysis and statistical sequence modeling.

(Oudot, 2003) developed a very large lexicon (200,000 words) omni-user system that includes writer adaptation component. It is based on the activation-verification model in perceptive psychology field. Encoding experts of the input signal, extract probabilistic information at different levels of abstraction (geometrical and morphological) while neuronal expert of segmentation generates a trellis of segmentation hypotheses. The trellis is explored by a probabilistic fusion engine that uses information of the encoding experts and the lexicon in order to provide the best transcription of the input signal.

2.4.2 Developments in Offline Handwriting Recognition

Table 2.2 summarizes various works done in offline handwriting recognition. The method used, the size of the lexicon used and the types of system it can handle are given. Most techniques that are applied to online recognition are also applicable to offline recognition. As can be observed from the table, HMM and hybrid of HMM

and NN are popular. Dynamic programming is also widely applied. Most researchers handled small lexicon while a few caters for large lexicon size of more than 10,000 words.

Table 2.2 Offline handwritten word recognition systems

Author	Method	Lexicon Size	Comments
(Burges, 1993)	DP/NN	1,000	UNC,OMNI
(Cai, 1993)	DP/Fuzzy	14	UNC
(Cho, 1994)	HMM	10,000	CUR, OMNI
(Chen, 1994)	HMM	271	UNC, OMNI
(Gader, 1994)	NN/DP	100	UNC, OMNI
(Chen, 1995)	HMM	1,000	UNC, OMNI
(Bunke, 1995)	HMM	150	CUR, WD, 5 Writers
(Guillevic, 1995)	HMM/kNN	30	UNC, OMNI
(Gader, 1995)	DP	746	HAND, OMNI
(Mohamed, 1996)	DP	100	UNC, OMNI
(Kim, 1997)	DP	1,000	UNC, OMNI
(Farouz, 1998)	HMM/NN	1,000	UNC, OMNI
(Dzuba, 1998)	DP	40,000	CUR, OMNI
(Augustin, 1998)	HMM/NN	28	CUR, OMNI
(Lallican, 1999)	HMM/ODREC	197	CUR, OMNI
(Madhvanath, 1999)	DP	1,000	CUR, OMNI
(Saon, 1999)	HMM	26	UNC, OMNI
(Bippus, 1999)	HMM	400	UNC, OMNI
(Procter, 2000)	HMM	713	CUR, WD, 1 Writer
(Mohamed, 2000)	HMM/Fuzzy	100	UNC, OMNI
(Scagliola, 2000)	DP	1,000	CUR, OMNI
(Marti, 2000)	HMM	7,719	UNC, OMNI, 250 Writers
(Brakensiek, 2000)	HMM	30,000	CUR, WD, 4 Writers
(Favata, 2001)	DP	1,000	UNC, OMNI
(Tay, 2002)	HMM/NN	197	CUR, OMNI

UNC: Unconstrained, OMNI: Omniwriter, CUR: Cursive,
WD: Writer–Dependent HAND: Handprinted

2.4.3 Issues in Preprocessing

In online handwriting recognition system, preprocessing of the trajectory of input pattern facilitates the description of the input signal and improves the quality of the description as well. Preprocessing tasks of online character patterns includes; noise elimination, data reduction and signal normalization. Noise in handwriting

input signal can be due to users' erratic hand motions and the imperfection in the process of digitization of the signal. The forms of noise elimination or more realistically, noise reduction are signal smoothing, filtering, wild point correction, and dot reduction. Wild point reduction can replace or eliminate occasional spurious points and dot reduction reduces dots to single points. As input devices quality improves, trajectory noise is less of a problem and normally smoothing only, will be sufficient.

Data reduction can be accomplished by two approaches: equidistance sampling or line approximation (feature point detection). With equidistance sampling, the trajectory points are resampled so that adjacent points are of equal distance. Equidistance resampling however does not reduce the data very much. A better data reduction rate can be achieved by detecting feature points. Feature points are the corner points and the ends of a stroke trajectory. The idea is to estimate the curvature at each point on the curve and retain the points of high curvature Complementary to detecting feature points, polygonal approximation, which recursively finds the vertex of maximum point-to-chord distance is also useful to reduce data representation and achieve better performance. Approximation of strokes by line is also popularly used to reduce data signal in many online recognition systems. The latter two are more popular in Chinese character based recognition. (Liu, 2004).

Signal normalization is a standard procedure in almost every recognition system. Normalization can be linear or nonlinear. In linear normalization - the coordinates of stroke points are shifted and scaled such that all points are enclosed in a standard box. Another option in linear normalization is to use moment normalization, where the centroid of input pattern is shifted to the center of standard box and the second-order moments are scaled to a standard value. As for nonlinear normalization, in offline signals, coordinates of stroke points are reassigned according to the line density distribution. The aim is to equalize the stroke spacing. For online signals, the line density can be computed directly from the online trajectory. Moment normalization yield comparable recognition accuracy to nonlinear normalization. (Beigi, 1994) describes methods for size normalization which have then been adapted by many authors. Referring to Figure 2.2 and Figure 2.3, a handwritten

word can be imagined to be written inside the four reference lines; the descender line at the bottom of the small letter “g”, the base line at the bottom of any small letter, the core line at the top of a small letter and the ascender line at the top of the capital letter. To perform size normalization, the baseline and the core line need to be estimated. The area between the two lines is always non empty for any letter and reliable for any size normalization. Once an estimate for this area is obtained, a magnification factor can be computed from the ratio of this area and the input to be used to normalize the input signal. For other non Latin based languages, these are not applicable and the full height of the writing determines the magnification factor. Before size normalization can be done, slope correction needs to be done to align the writing with horizontal axis.

An important aspect of preprocessing is delayed strokes processing (Hu, 1996). Delayed strokes refer to strokes such as the cross of “t” and “x” as well as the dots for “i” and “j”. The crosses are normally written last, thus called delayed stroke, which normally separates from the main body of the letter. In most online handwriting systems, delayed strokes are first detected in preprocessing and then either discarded or used later. If they are used, they are treated as special letters in the alphabet. A word with delayed strokes is given alternative spellings to accommodate different sequences with delayed strokes written in different order.

2.4.4 Issues in Segmentation Stage

In small vocabulary handwriting recognition, segmentation is not an issue as holistic method is normally used. As the size of the vocabulary gets larger, analytical approach is more preferred. It involves segmentation of the handwriting into primitives such as strokes, pseudo-letters or letters. Segmentation issue have not been solved or even addressed fully. What kind of primitives to use and the methods to segment them is heuristically based on experience.

Segmentation is the operation that seeks to decompose a word signal to a sequence of sub signals that contains isolated characters. Segmentation points can be at the extreme points in x and y axes, cusps or sharp corners, critical points or

multiple points. Segmentation is a critical phase of the single word recognition process. This is proven by the fact that character recognizer trained using isolated characters is better than character recognition trained using segmented characters from cursive words since isolated characters are "naturally segmented" by the writer. There are two main strategies for segmentation: (a) straight segmentation or referred to also as classical analytical approach to segmentation and (b) recognition-based segmentation or referred to also as SegRec approach (Tay, 2002). Straight segmentation tries to decompose the signal into a set of sub signal, each one corresponding to a character. Straight segmentation is difficult as it needs to perform some complex analysis. It is suitable only for tasks like segmentation of typewritten or hand printed words and thus may not be very appropriate to be applied for cursive or unconstrained handwritings.

SegRec approach subdivides the word signal into a set of sub signals whose combinations are used to generate character candidates. The number of sub signals is greater than the number of characters in the word and the process is referred to also as over segmentation. During recognition, sub signals are combined to form character hypothesis. Character hypothesis are evaluated and combined to form a word which will be compared to each available word in the lexicon. Because words are formed by concatenation of smaller units such as characters, it is very easy to enlarge the word lexicon. As can be seen, this lexicon-driven system involves tightly coupled segmentation and recognition process. Recognition-based segmentation driven by the lexicon solves the complex situation of having to first know the correct word in order to segment it and having to first segment the word correctly in order to be able to recognize it, a situation called Sayre's paradox (Steinherz, 1999) mentioned earlier. The quality of the over segmentation process depends on whether or not we missed the detections of ligatures and the ratio of the number of primitive sub signals produced and the number of characters in the word. The optimal segmentation is determined by the optimal combination of sub signals in forming character hypothesis.

(Bengio, 1995b) and (Tay, 2002) describes 2 methods of over segmentations in a word recognition system; the input segmentation (INSEG) and output segmentation (OUTSEG). Figure 2.4 compares the two segmentation approaches using the offline

word “clock”. In INSEG, over segmented sub signals are combined heuristically and recognized as characters that are combined optimally by a decoder using dynamic programming feature of the HMM to produce the word score. The size of each sub signal or cut is not fixed or uniform. They are based on the dynamics of the pen such as pen lift and pen velocity as well as geometrical clues such as spaces and corners. In OUTSEG, entire word is accepted by the recognizer. The input word is cut or rather divided into overlapping cut windows of the same size. Each window will be separately recognized. The segmentation decisions are delayed until after the recognition. A sequence of scores for each character at each location in the input is produced. The HMM in the recognizer models the sequential structure of the word while a character recognizer (normally the convolution neural networks) spots and classify the characters.

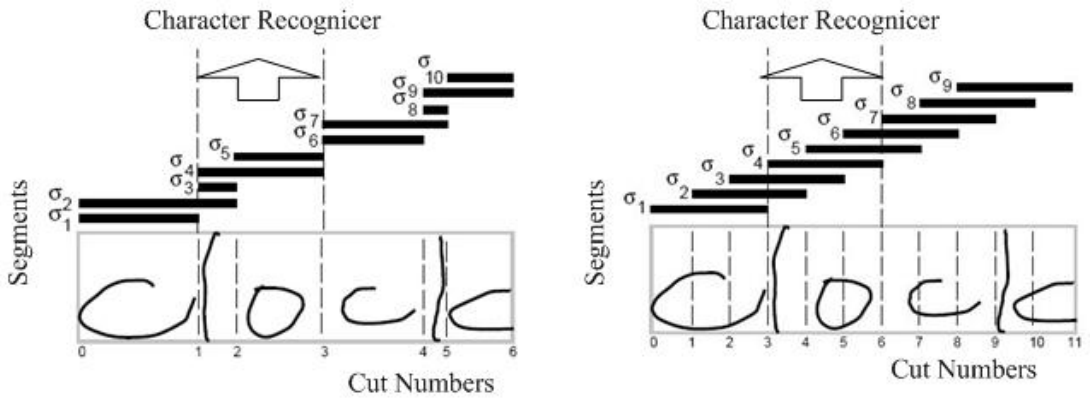


Figure 2.4 (a) *INSEG* based segmentation (left) showing 3 hypothesis σ_3, σ_4 and σ_5 for INSEG method which are within slices 1-2 and 2-3. (b) *OUTSEG* based segmentation (right) which shows segment σ_4 within window 3-6 and overlapping windows σ_5 and σ_6

2.4.5 Issues in Word Recognition

Word recognition method depends on whether the recognition involves segmentation or not. In holistic or non segmentation based system, the whole world is recognized in one go. Normally, recognition of a word involves calculation of some similarity or distance measure between the features of the word and the words

contained in the lexicon. Methods such as the nearest neighbor or k-nearest neighbor can be used in the recognition where the minimum distance measure determines the class of words. HMM has also been used, especially for system with a small lexicon size. A useful function of this holistic word recognition is as a preprocessor of the segmentation based recognizer where it can be used to preselect a small subset of probable words. This in a way reduce the lexicon size before the segmentation based recognition performs more detailed recognition of each word in the reduced lexicon (Madhvanath, 2000). This also reduced the recognition time. Holistic recognition is quite robust. Its performance is not affected too much even if we have deformed handwriting. Thus it can also be used in the design of combination of multiple recognizers.

As mentioned, segmentation based recognition can be divided into straight segmentation and SegRec. In straight segmentation based system, once the segmentation into characters has been done, the recognition is a straight forward step. A character recognizer trained on isolated character can be used without much change to produce recognition score for each character that have been segmented. NN has been a popular character recognizer for this approach. SVM can be one as well. The recognized word is formed by concatenating the characters.

In SegRec approach, a character recognizer is used to recognize many hypothesis characters made up of combination of the smaller over segmented segments. The word recognizer combines the most suitable combination of hypothesis that gives the highest word score among all words in the lexicon. The best word score eventually determine the ultimate segmentation points of the word. As mentioned earlier, there are two ways to segment words in recognition-based segmentation; (a) the OUTSEG method and (b) the INSEG method.

OUTSEG or Output Space Segmentation approach allows segmentation points to be decided at the output space. Initially, the word signal is segmented implicitly into uniform size entities which can be overlapping, that is smaller or equal to character. Then, the recognition is carried out at the output space by associating groups of these smaller entities to form a particular character in a word. INSEG or Input Space Segmentation on the other hand requires segmentation points to be decided explicitly

by using spatial information in the word. All possible segmentation points are determined and cuts are made at these points. Then the cuts are combined into character hypotheses and passed to character recognizer. These character hypotheses can represent part of a character, a full character, a few characters, or part of a character combined with part of another characters. This is a big challenge for the character classifier, which has to deal with unseen patterns during training stage. Actually, they are outliers and classifiers such as NN behave inconsistently in this situation. The recognition process involves selecting only the character hypotheses that represents actual character signals forming a particular word.

Normally, OUTSEG recognition method involves using convolutional neural network (CNN) and hidden Markov model in a hybrid system where the CNN spot and recognize the characters and the HMM perform the sequential modeling for the word. The CNN can be a Time delay Neural network (TDNN) used in online handwriting recognition or space displacement neural network (SDNN) for off-line handwriting recognition. These NN have some shift independent capabilities allowing to detect a character even when it is not currently centered in the window.

For INSEG approach, the character recognizer can be based on a probability distribution function (PDF) classifier in the global HMM itself. This is the case of a discrete HMM. Sometimes, a continuous density HMM with Gaussian distribution function is used instead. The continuous HMM is proven to give better recognition result compared to discrete HMM. PDF classifiers are not discriminant since its training only involves in class data, without taking into account the out of class data. To obtain a discriminant classifier, classifier giving output as posterior probability can be used. The most popular posterior probability based character recognizer used in this approach is the ANN trained with Back propagation (BP). In BP ANN, data from all classes are involved during training which makes it discriminant in nature thus able to better discriminate among classes. However, by using posterior probability based classifier, the output needs to be normalized because prior probability is embedded in the posterior (refer to Bayes theorem) resulting in a biased recognition. Normalizing the posterior probability output gives likelihood or similar probability. Besides ANN, other discriminant classifiers that give posterior

probability or likelihood score as the output at the character level can be used in an INSEG based word recognition system.

The final step in handwriting recognition is to compare the handwritten word that is being recognized with the reference patterns to determine their similarity to decide which pattern or model best represents the word being recognized. In this step also, the best segmentation of the word into characters is determined. Language models may be used at this level. A few methods of implementing language model that are commonly used are lexicon trie or simple Markov model or n-gram (Guyon, 1996). Lexicon trees store list of words with their frequency while character n-grams are used to predict the next character given a window of (n – 1) past characters. In sentence recognition, word n-grams are used to predict the next word given a window of (n – 1) past words.

SegRec word recognition is essentially a best path problem that incorporates character classification scores, segmentation information and the language model. Character classification score can be pure probability values or negative log probabilities. The overall score for a path in the graph is given by the product of the character score of the arcs traversed. The probability of a given word is given by summing over all possible ways of character combinations to produce that word. The most probable word is the recognition result. The forward algorithm described in HMM is an efficient dynamic programming (DP) technique to compute the above sum. The Viterbi algorithm which picks the best single path in the graph as the recognition is often used to approximate the forward algorithm, for computational reasons. This means replacing the sum in equation with the largest term to make this approximation. Approximate search procedures to find the most probable word are often preferable for computational reasons. These include beam search procedures (Ney, 1987) and the A* algorithm (Soong, 1991) and various fast match techniques can be used to narrow down the search space.

2.4.6 Issues in Post Processing Stage

Post processing stage may involve using search strategies and verification approaches that allow for achieving faster recognition and improvement in the

accuracy. Speed improvement can be obtained by various methods in the search techniques such as lexical tree search, standard and constrained lexicon–driven level building algorithms, two–level decoding algorithm, and a distributed recognition scheme. The recognition accuracy can be improved by post–processing the list of the candidate N–best–scoring word hypotheses generated by the baseline recognition system. The list also contains the segmentation of such word hypotheses into characters.

Verification module can be used to generate a score for each segmented character and in the end; the scores from the baseline recognition system and the verification module are combined to optimize performance. A rejection mechanism can be introduced over the combination of the baseline recognition system with the verification module to improve significantly the word recognition.

2.5 SVM in Speech and Handwriting Recognition

This section reviews current usage of SVM from the perspective of handwriting recognition. A few references may be made to its usage in speech recognition as its application is similar to the one in handwriting recognition. In general, SVM have been used in handwriting recognition in a number of ways; as a standalone recognizer in a fixed feature based character recognition system (Ahmad, 2004a), as a replacement of HMM in a sequence processing based character or word recognition system (Bahlmann, 2002) or as a final decider in the final output of a handwriting recognition system. The author describe in the following subsections the various ways of SVM's usage.

2.5.1 SVM in Speech Recognition

(Ganapathiraju, 2002) and (Ganapathiraju, 2004) describes application of SVM in a large vocabulary speech recognition. SVM is used in a hybrid HMM/SVM setting. Since SVMs is inherently a static classifier and HMMs have the ability to handle dynamic data, the two complements each other. An important issue that had to be addressed in this hybrid system is the fact that normally, SVMs output a

distance measure, while the Viterbi decoding algorithm typically uses likelihoods or posterior probabilities. Therefore SVM distances outputs are converted to posterior probabilities.

2.5.2 SVM with DTW Kernel in Character Recognition

(Bahlmann, 2002) describes an approach for on-line character recognition that combines dynamic time warping (DTW) and support vector machines (SVMs) by establishing a new SVM kernel. He called the kernel - Gaussian DTW (GDTW) kernel and his method as SVMGDTW. The kernel approach has an advantage over common HMM techniques because it does not model generative class conditional densities. Instead, it directly addresses the problem of discrimination by creating class boundaries and does not have modeling assumptions. By incorporating DTW in the kernel function, general classification problems with variable-sized sequential data can be handled. SVMGTDW method can in fact be applied to other similar problems such as speech recognition. Bahlmann compared his kernel approach to an HMM based technique on the UNIPEN handwriting database and showed that he achieve comparable results.

In SVM research, work on kernels for sequential data has been done by (Jaakkola, 1999) and (Watkins, 2000). Jaakkola developed an SVM kernel in their application of protein homology detection and refer to it as Fisher kernel. Watkins developed several explicit kernels for sequential data and shows that they are proper SVM kernels under certain conditions. However, the kernels mentioned above are still based on an estimation of generative parameters. The GDTW kernel on the other hand presumes less model knowledge and is less complex. Comparing GDTW kernel to HMM-based classifier on the UNIPEN data shows that recognition rate is better for relatively small training sets but comparable for larger training sets.

2.5.3 SVM as a Character Recognizer in a Hybrid System

(Camastra, 2007) describes a cursive character recognizer as a module in an offline cursive word recognition system based on a segmentation and recognition

approach. The character classification is done by using Support Vector Machines (SVMs) and a Neural Gas. The Neural Gas is used to verify whether lower and upper case version of a certain letter can be joined in a single class or not. Once this is done for every letter, the character recognition is performed by SVMs.

2.5.4 SVM in Multiple Classifier Methods

To achieve an optimal recognition rate, many researches use different methods for combining multiple classifiers to compensate the weakness of one classifier, by the strength of the other classifiers. The combination method can use Local Accuracy Estimates, Local Learning Algorithm, Adaptive Mixtures of Local Experts or aggregation of the decisions obtained from individual classifiers to derive the best final decisions from a statistical point of view. The disadvantage of most of these methods is the complexity of optimization for each classifier and the definition of local area in terms of K-nearest neighbors which requires storing in the system memory all the training examples. These constraints are prohibitive in real handwriting recognition systems where some training sets can contain large number of examples.

(Bellili, 2000) uses a combination of multilayer perceptron (MLP) neural network and SVM classifiers. The SVMs are used to improve the performances of an MLP based digit recognizer. The hybrid SVM/MLP architecture is based on the idea that the correct digit class of the recognizer almost systematically belongs to the two maximum MLP outputs and that some pairs of digit classes constitute the majority of the recognizer errors. Specialized local SVMs are introduced to detect the correct class among these two classification hypotheses. The hybrid MLP-SVM recognizer achieves a recognition rate of 98.1%, for real mail zipcode digits recognition task, a performance better than several classifiers reported in recent researches.

2.5.5 SVM in Non Roman Handwriting Recognition

Support vector machines have also been observed to achieve reasonable generalization accuracy for non-Roman handwriting recognition such Thai

(Sanguansat, 2004) Arabic (Bentounsi, 2004) and Devanagari/Telugu scripts (Chakravarthy, 2007).

(Sanguansat, 2004) proposed a method for online Thai handwritten character recognition using HMMs and SVMs with a generalized Fisher kernels (called score-space kernels) based on underlying generative models. In the first phase, HMMs are used for multi-classification, then SVMs are applied to resolve any uncertainty remaining after the first-pass HMM-based recognizer (on certain classes only because the results of some classes are worse). Confusion matrix of the HMM-based recognizer is used to find the confused candidates in each class. If there is one candidate, it means there is no confusion in this class and HMMs alone are sufficient to classify. If there is more than one candidate, SVMs are applied. If there are more than two, the multi-class method is applied. Symmetric likelihood ratio score-space was proposed where one observation sequence is mapped to only one score-vector. Experimental results show the average recognition rate improved from 89.9%, using baseline HMM, to 92.5%, using SVM with score space kernel.

(Chakravarthy, 2007) uses SVM for online handwritten character recognition for Indian scripts. A number of separate feature vector combinations were used and compared. Features compared are stroke points, Fourier series coefficient and spatio-structural features (shape feature), Hilbert transform, stroke points appended with stroke velocity, PCA based feature and Fisher linear discriminant (FLD) based feature vector. The standard gaussian kernel is used for training. Multiple classifiers approach were also taken where (1) the class corresponding to the maximum of normalized value among all the classifiers is selected as the best representative for the given test sample, (2) Majority vote is applied on the top K-output values from each classifier, (3) Normalized output value from each classifier is selected and concatenated and passed to another SVM based classifier.

2.6 Summary

In this chapter, we review pattern recognition and speech recognition as the lead towards describing in detail the reviews on state of the art for handwriting

recognition. We review both offline and online handwriting recognition with respect to the various stages in the recognition system; preprocessing, segmentation, feature extraction, recognition and post processing. Finally, a review of the usage of SVM in speech and handwriting recognition is given.

CHAPTER 3

HIDDEN MARKOV MODEL

3.1 Introduction

Handwritings are collection of signals captured by appropriate devices. Thus, as with any signal, they can be described theoretically by using a signal model. The model can be used is two ways; (a) to describe the process of writing, given a signal that gives some desired output, (b) to learn about the signal source by simulation without the source being available. Signal models are either deterministic or non-deterministic (statistical). The differences between the two are that deterministic models use some known properties of the signal and only certain parameters need to be determined while non-deterministic or statistical models determines the statistical properties of the signal assuming that it can be characterized as a parametric random process such as Gaussian, Poisson or Markov processes. The signal is assumed to be well characterized and its related parameters can be determined or estimated in a precise and well-defined manner.

Hidden Markov Model (HMM) is a statistical model of Markov process. It is rich in mathematical structures which can be used to model signals in real applications. An HMM is a variable-size collection of random variables with an appropriate set of conditional independence properties. Informally, an HMM is a variant of a finite state automata (FSA), which model a behavior composed of states, transitions and actions. However, HMM, unlike FSA, are not deterministic. A

normal FSA emits a deterministic symbol in a given state. Further, it also has deterministic transitions to another state. A stochastic FSA has either one of emission or transition which is probabilistic. HMM, on the other hand is doubly stochastic, both in the transition and emission. Given an FSA to model a string of symbols it can be easily determined if the string has been generated by the FSA and if it is what the sequence of state transitions undertaken was (Boulard, 2003). With an HMM, the first stochastic process is represented by the probability that the HMM generated the string and the second by the sequence of state transitions undertaken which is “hidden", hence the name Hidden Markov Model. The stochastic emission models the local properties and the stochastic transition models the sequential properties.

Early theory of HMM was published by Leonard E. Baum and other authors (Baum, 1970). It has been successfully used to address complex sequential pattern recognition problems, among them continuous speech processing and recognition, cursive handwriting recognition, time series prediction and biological sequence analysis (Boulard, 2003). Its first usage was in speech processing as reported by (Baker, 1975) and (Jelinek, 1976). The usage was further popularized in speech recognition in the 80s by (Levinson, 1983) and (Rabiner, 1986a). During this time, several HMM-based speech recognition systems from AT&T, BBN, and CMU showed superior results (Chow, 1987) (Lee, 1988). The success of these systems dramatically increased interest in applying HMMs to speech recognition and other difficult pattern recognition problems such as handwriting recognition.

Some usages of HMM in handwriting recognition can be traced in the following papers by (Nag, 1986), (Kundu, 1988), (Matan, 1992a) (Ha, 1993), (Schenkel, 1993), (Schenkel, 1995) and (Bengio, 1995a). This chapter introduces HMM and its usage in handwriting recognition.

3.2 Theory of HMM

There are two types of HMMs classified by their observation probability densities: discrete-density HMMs and continuous-density HMMs. For simplicity, the

discussion here will be limited to discrete-density HMMs. A more detailed explanation of HMMs can be found in (Rabiner, 1993), (Huang, 1990) and (Lee, 1988).

As mentioned earlier, HMM is a statistical model of Markov processes. To understand discrete-density HMMs, a review of the discrete-state Markov process is necessary.

3.2.1 Discrete-State Markov Process

A Markov process is a stochastic process that satisfies the Markov condition in which its future behavior depends only on its present state, not on the past. It is also called a memory less system. A discrete-state Markov process can be in one of a set of N distinct discrete states, $S_1, S_2 \ldots S_N$ at any given time. Let Q_n denote the state of the process at time n. The probability of the process being in state S_i at time n is denoted by $P(Q_n = S_i)$.

Markov condition implies state-independence assumption. Simply stated, the present state depends only on the previous state. It can be formally stated as follows:

$$P(Q_n = S_i \mid Q_{n-1} = S_i, Q_{n-2} = S_a, \ldots, Q_0 = S_b) = P(Q_n = S_j \mid Q_{n-1} = S_i) \quad \text{(Eq. 3.1)}$$

$$\forall i, j, a, b \text{ and } n$$

Since a discrete-state Markov process satisfies the Markov condition, the initial state probabilities and the state transition probabilities from one state to the next together characterize the process completely. The probabilities of starting in a particular state or the initial state probabilities are denoted by $\Pi = \{\pi_i\}$ where

$$\pi_i = P(Q_0 = S_i) \qquad 1 \le i \le N \quad \text{(Eq. 3.2)}$$

$$\text{with } \sum_i \pi_i = 1$$

The state transition probabilities are denoted $A = \{a_{ij}\}$, where:

$$a_{ij} = P(Q_n = S_j \mid Q_{n-1} = S_i) \qquad \forall \ 1 \le i \le N \text{ and} \quad \text{(Eq. 3.3)}$$

$$\sum_i a_{ij} = 1 \quad \text{(Eq. 3.4)}$$

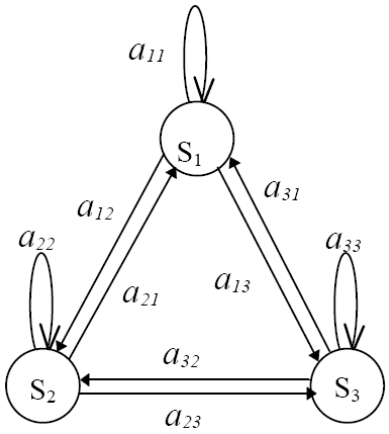


Figure 3.1 A 3-state markov process

As an example, a Markov process with 3 states (S_1, S_2 and S_3) is shown in Figure 3.1. The circles are the states and the arrows indicate the transitions that are possible between the states. In the diagram, a_{ij} is the state transition probability from state i to state j and π_i is the initial state probability from state i.

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \text{and} \tag{Eq. 3.5}$$

$$\Pi = \{\pi_k\} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \tag{Eq. 3.6}$$

The duple $\{A, \Pi\}$, completely parameterizes the discrete-state Markov process.

3.2.2 Extending Discrete-State Markov Processes to Hidden Markov Models

If we extend the discrete-state Markov process so that there is a non-deterministic (or probabilistic) observation associated with each state, we have a Hidden Markov Model (HMM). Here, we assume that there is a symbol O_i that is observed when the process is in the state i, according to some probability. Thus, there is a sequence of observations that is observed and there are many possible state

sequences which generate an observation sequence. The state sequence however is hidden.

A formal definition for HMM is as follows. Let the number of distinct observation symbols that can be emitted in each state be M. Let O_n be the observation at time n and the event for which the observation symbol is k be denoted by v_k. The state observation probabilities is denoted as $B = \{b_{iv_k}\}$ where

$$b_{iv_k} = P(O_n = v_k \mid Q_n = S_i) \qquad 1 \le i \le N \text{, and} 1 \le k \le M \tag{Eq. 3.7}$$

$$\sum_k b_{iv_k} = 1$$

Since HMM satisfies the output independence assumption, the probability of present observation given past observations, depends only on the current state. As such, we have

$$P(O_n = v_k \mid O_{n-1} = v_a, O_{n-2} = v_b \ldots, = Q_0 = v_c, Q_n = S_k) = P(O_n = v_k \mid Q_n = S_k) \quad \forall i,j,a,b \text{ and } n \tag{Eq. 3.8}$$

The triple, $\{A, B, \Pi\}$, normally denoted together by λ completely parameterizes an HMM. i.e., $\lambda = \{A, B, \Pi\}$ and Π is the initial state probabilities.

Figure 3.2 shows a simple example of discrete density HMM with 3 states and 2 observation symbols, i.e.: N = 3 and M = 2. The state transition probabilities are:

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \text{and} \tag{Eq. 3.9}$$

the state observation probabilities for the two symbols $\{1,2\}$ are:

$$B = \{b_{iv_k}\} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \quad \text{and} \tag{Eq. 3.10}$$

The initial state probabilities are:

$$\Pi = \{\pi_k\} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \quad \text{(Eq. 3.11)}$$

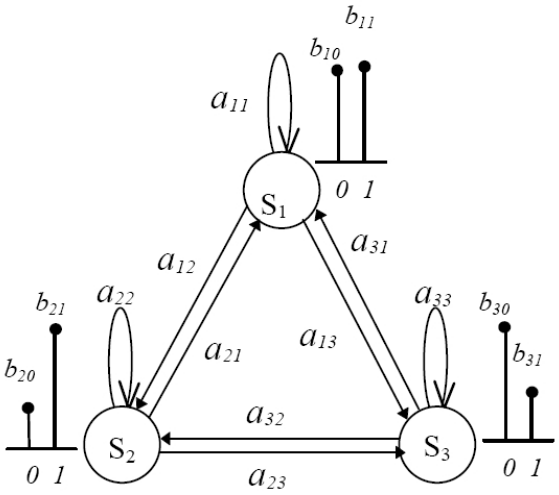


Figure 3.2 A 3-state HMM with 2 observation symbols {0, 1}

3.2.3 Three Problems of HMM

Given the form of the HMM discussed in the previous section, there are three key problems of interest that must be solved for the model to be useful in real world applications. These problems are the following:

(a) Given the observation sequence $O = O_1O_2...O_T$ and the HMM model $\lambda=\{A, B, \Pi\}$, how to compute $P(O|\lambda)$, the probability of the observation sequence. This is the *Evaluation Problem*.

(b) Given the observation sequence $O = O_1O_2...O_T$ and the HMM model $\lambda=\{A, B, \Pi\}$ how to choose a state sequence $Q = Q_1Q_2...Q_T$ which maximizes $P(Q, O|\lambda)$. This is the *Decoding Problem*.

(c) Given the observation sequence $O = O_1O_2...O_T$ and the HMM model $\lambda=\{A, B, \Pi\}$, how to adjust the model parameters $\lambda=\{A, B, \Pi\}$ which maximize $P(Q, O|\lambda)$. This is the *Training Problem*.

In (a), given a model and a sequence of observations, we compute the probability that the observed sequence was produced by the model or another words we evaluate the model. ie: by comparing the model with competing models, we can choose which is the best match to the observations. In (b), we attempt to uncover the state sequence (hidden part of the model). Using an optimality criterion, the best state sequence which is found can be used to learn about the structure of the model, and to get average statistics, behavior, etc. within individual states. In (c), we attempt to optimize the model parameters so as to best describe how the observed sequence comes about. The training problem is crucial since it allows to optimally adapt model parameters to observed training data to create best models for real phenomena.

3.2.4 A Solution to the Evaluation Problem – The Forward Algorithm

The evaluation problem is to compute $P(O|\lambda)$, the probability of the observation sequence, $O = O_1O_2...O_T$, given the model parameter λ. Since the state sequence, $Q = Q_1Q_2...Q_T$, corresponding to the observation sequence O is hidden, $P(O|\lambda)$ has to be computed by summing $P(O, Q|\lambda)$ over all possible state sequences.

$$P(O \mid \lambda) = \sum_{allQ} P(O,Q \mid \lambda) \qquad \text{(Eq. 3.12)}$$

where $P(O,Q \mid \lambda) = P(O|Q, \lambda)\ P(Q \mid \lambda)$ (Eq. 3.13)

The state independence assumption (Eq. 3.1), allows us to write:

$$P(Q \mid \lambda) = \pi_{Q1} a_{Q_2Q_1} a_{Q_3Q_2} .. a_{Q_TQ_{T-1}} \qquad \text{(Eq. 3.14)}$$

Also, the output independence assumption allows us to write :

$$P(O,Q \mid \lambda) = b_{Q_1O_1} b_{Q_2O_2} .. b_{Q_TO_T} \qquad \text{(Eq. 3.15)}$$

Therefore,

$$P(O \mid \lambda) = \sum_{allQ} \pi_{Q_1} (a_{Q_2Q_1} a_{Q_3Q_2} .. a_{Q_TQ_{T-1}}) (b_{Q_1O_1} b_{Q_2O_2} .. b_{Q_TO_T}) \qquad \text{(Eq. 3.16)}$$

The direct calculation of $P(O|\lambda)$ in (Eq. 3.16) involves calculations on the order of $2TN^T$. This computation becomes unfeasible as the number of possible states, N, or the length of the observation sequence T increases. This necessitates a more efficient way of computing $P(O|\lambda)$. Fortunately, an efficient algorithm called *Forward-Backward* algorithm exists. First, let us define the forward variable:

$$\alpha_t(i) = P(O_1 O_2 \ldots O_t, Q_t = S_i | \lambda) \qquad \text{(Eq. 3.17)}$$

The variable $\alpha_t(i)$ denotes the joint probability of the partial observation sequence, $O_1 O_2 \ldots O_t$, and the state S_i at time t, given the model λ. It can be calculated recursively:

$$\alpha_t(i) = \begin{cases} \pi_i b_{io_1} & t = 1, \quad 1 \le i \le N \\ \sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji} b_{io_t} & 2 \le t \le T \quad 1 \le i \le N \end{cases} \qquad \text{(Eq. 3.18)}$$

From the definition of the forward variable, it is observed that the probability of the entire sequence can be expressed as:

$$P(Q \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \qquad \text{(Eq. 3.19)}$$

(Eq. 3.17) to (Eq. 3.19) illustrate how to compute $P(O|\lambda)$ by first recursively evaluating the forward variables, $\alpha_t(i)$, from $t = 1$ to $t = T$ and then summing all the forward variables at time T, the $\alpha_T(i)$'s. The above steps are often referred to as the forward algorithm. The number of calculations involved is on the order of TN^2 instead of $2TN^T$. Hence, the forward algorithm can be used to solve the evaluation problem much more efficiently.

3.2.5 A Solution to the Decoding Problem – The Viterbi Algorithm

The decoding problem involves finding an optimal state sequence given the observation sequence, $O = O_1 O_2 \ldots O_T$, and the model parameter λ. The optimality criterion is to maximize $P(Q,O|\lambda)$ which is the joint probability of the state sequence, $Q = Q_1 Q_2 \ldots Q_T$, and the observation sequence $O = O_1 O_2 \ldots O_T$, given the model λ.

The optimal state sequence is denoted by Q^*. Viterbi algorithm which is a popular algorithm based on dynamic programming, can be used to solve this optimization problem. We use $\delta_t(i)$ to denote the maximum probability of the optimal partial state sequence, $Q_1Q_2...Q_{t-1}$, with the state S_i at time t and observing the partial observation sequence, $O_1O_2...O_t$, given the model λ.

$$\delta_t(i) = \max_{Q_1Q_2\cdots Q_{t-1}} P(Q_1Q_2\ldots Q_{t-1}\, Q_t = S_i,\, O_1O_2\ldots O_t \mid \lambda) \qquad \text{(Eq. 3.20)}$$

Similar to the forward variable $\alpha_t(i)$, $\delta_t(i)$ can be calculated recursively as follows:

$$\delta_t(i) = \begin{cases} \pi_i b_{io_1} & t = 1, \quad 1 \le i \le N \\ \max_{1\le i\le N} \delta_{t-1}(j) a_{ji} b_{io_t} & 2 \le t \le T \quad 1 \le i \le N \end{cases} \qquad \text{(Eq. 3.21)}$$

From the definition of $\delta_t(i)$, it is clear that :

$$P(Q^*, O|\lambda) = \max_{1\le i\le N} \delta_T(i) \qquad \text{(Eq. 3.22)}$$

Using (Eq. 3.21) and (Eq. 3.22), we can compute the joint probability of the optimal state sequence and the observation sequence given the model, $P(Q^*, O|\lambda)$. Note that the memory usage is very efficient, i.e., at any time t, only N forward variables, $\delta_t(i)$ need to be stored. By keeping track of the argument i in both equations as $P(Q, O|\lambda)$ is being maximized, we can recover the optimal state sequence completely.

Also note that $P(Q^*, O|\lambda)$ can be viewed as the biggest component of $P(O|\lambda)$ in (Eq. 3.12). When $P(Q^*, O|\lambda)$ is a good approximation of $P(O|\lambda)$, we can use the Viterbi algorithm instead of the forward algorithm for the evaluation problem. This will conserve computation. Since the computational complexity of the Viterbi algorithm is even less than that of the forward algorithm.

3.2.6 A Solution to the Training Problem – The Baum-Welch Algorithm

The training problem is by far the most difficult of the three basic problems. The training problem computes the optimal model parameter, λ, given an observation sequence, $O = O_1O_2...O_T$. Here, the optimality criterion is to maximize $P(O|\lambda)$, the probability of the observation sequence given the model λ. Generally we expect the optimal model to have the same number of states and observations. Intuitively, we want to think of training an HMM as methods for making slight adjustments to an already somewhat-working model.

There is no known analytical solution that exists for the learning problem. There are however, popular iterative algorithms for addressing it: the Baum-Welch Algorithm, and Viterbi Training. In this section, we will focus on Baum-Welch Algorithm exclusively. The iterative procedures guarantee a locally optimal solution to the training problem. The Baum-Welch algorithm is a generalized expectation-maximization (EM) algorithm for finding maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only the observation training data. EM algorithm alternates between performing an expectation (E) step, which computes an expectation of the likelihood and maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated.

The two steps of the algorithm can be summarized as follows: (a) Calculating the forward probability and the backward probability for each HMM state; (b) On the basis of this, determining the frequency of the transition-observation pair values and dividing it by the probability of the entire string. This amounts to calculating the expected count of the particular transition-observation pair. Each time a particular transition is found, the value of the quotient of the transition divided by the probability of the entire string goes up, and this value can then be made the new value of the transition.

To discuss HMM training in detail, first, let us define the backward variable:

$$\beta_t(i) = P(O_{t+1}\, O_{t+2} \ldots O_T | Q_t = S_i, \lambda) \quad \text{(Eq. 3.23)}$$

The variable $\beta_t(i)$ denotes the probability of the partial observation sequence, $O_{t+1}\, O_{t+2} \ldots O_T$, given the state S_i at time t and the model λ. The backward variable is similar to the forward variable. It can also be calculated recursively:

$$\beta_t(i) = \begin{cases} 1 & t = T, \quad 1 \le i \le N \\ \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_{jo_{t+1}} & 1 \le t \le T-1 \quad 1 \le i \le N \end{cases} \quad \text{(Eq. 3.24)}$$

From the definition of the backward variable (Eq. 3.23) and the definition of the initial state probabilities (Eq. 3.2) it is clear that

$$P(Q \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \pi_i \quad \text{(Eq. 3.25)}$$

Second, let us define $\xi_t(i, j)$, the joint probability of the state S_i at time t and the state S_j at time $t+1$, given the observation sequence O and the model λ.

$$\xi_t(i, j) = P(Q_t = S_i, Q_{t+1} = S_j \mid O, \lambda) \quad \text{(Eq. 3.26)}$$

$\xi_t(i, j)$ can be completely expressed in terms of the forward variable, the backward variable, and the model λ.

$$\xi_t(i, j) = \frac{P(Q_t = S_i, Q_{t+1} = S_j, O \mid \lambda)}{P(O \mid \lambda)} \quad \text{(Eq. 3.27)}$$

$$= \frac{\alpha_t(i) a_{ij} b_{jO_{t+1}} \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_{jO_{t+1}} \beta_{t+1}(j)}$$

Note that the denominator of (Eq. 3.27) needs to be calculated only once. This quantity, which is equivalent to $P(O|\lambda)$, is often referred to as the alpha terminal. It indicates how well the model λ matches the observation sequence O. With the

current model as $\lambda = (A, B, \Pi)$, we can iteratively re-estimate the model, $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\Pi})$, where

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t-1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}, \quad \text{(Eq. 3.28)}$$

$$\overline{b_{iv_k}} = \frac{\sum_{t=1 \cap O_t = v_k}^{T} \sum_{j=1}^{N} \xi_t(i,j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \xi_t(i,j)} \quad \text{(Eq. 3.29)}$$

and

$$\overline{\pi_i} = \sum_{j=1}^{N} \xi_t(i,j) \quad \text{(Eq. 3.30)}$$

$\overline{a_{ij}}$ can be seen as the ratio of the expected number of transitions from state S_i to S_j to the expected number of transitions from state S_i to any state. Similarly, $\overline{b_{iv_k}}$ can be seen as the ratio of the expected number of times in state i while observing the symbol v_k to the expected number of times in state i. π_i can be seen as the expected number of times in state S_i at time $t = 1$. The above iterative procedure for updating the model λ is the essence of the Baum-Welch algorithm. Baum and others have proven that $P(O|\overline{\lambda}) \geq P(O|\lambda)$ for every iteration of the algorithm. Hence, $P(O|\overline{\lambda}) \approx P(O|\lambda)$ is used as the stopping criterion for the algorithm. The likelihood function, $P(O|\lambda)$ will eventually converge to a local maximum.

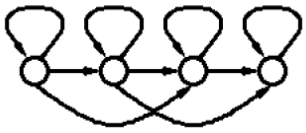
3.3 HMM Model Topology

In Baum-Welch algorithm, we refine an existing HMM so as to make it more suitable for a particular dataset. How to pick an initial HMM from scratch to match some empirical data is actually more of trial and error. We begin by making a "guess" at what a good model might be, and then use Baum-Welch to tune the

probabilities accordingly. The number of states and observations and the topology of the state transition graph need to be decided, which require some insight into the process being modeled. Normally, a few different models are tried before the best is picked.

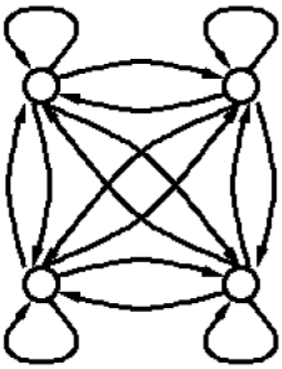
There exist many different HMM model topology. Figure 3.3 shows some example topologies for a 4 state HMM. Left-to-right model is a model which allow only left to right transition and does not allow backward transition while Ergodic model allows transition from a state to any other states. Linear model is a special case of left-to-right model without the skip between states while Bakis model is also a left-to-right model but allows a single state skip.

(a) linear model – transition allowed to the current or next state.



(b) Bakis model – transition allowed to the current, next or next 2 state.

(c) Left-to-right model – transition allowed to current and all states to the right



(d) Ergodic Model – transition allowed to current and any other states

Figure 3.3 HMM Model Topology

3.4 Using HMMs for On-line Handwriting Recognition

To use HMMs for solving handwriting recognition problem, we need to know how to model handwriting using HMM. Letters, words, and sentences can be modeled with HMMs. Building the model, ie: the training and the recognition of isolated words and sentences can be accomplished by using the solutions to the three basic HMM problems given in the previous section. In this section, the modeling of letters, words, and sentences are described respectively.

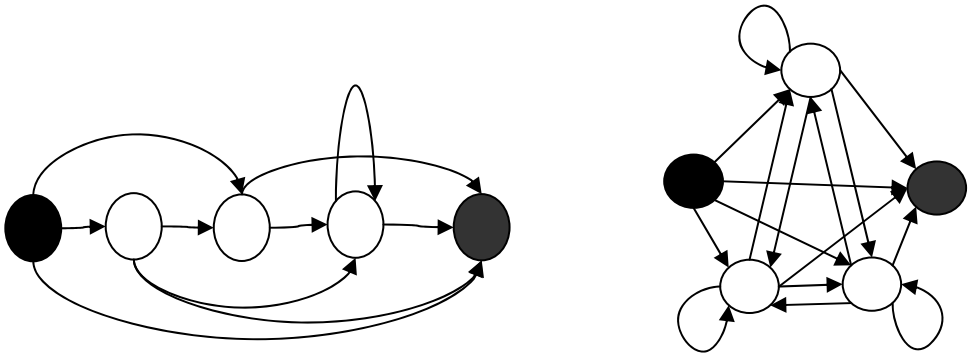
3.4.1 Modeling Letters

An HMM can model a letter. Normally a left-to-right HMM topology is used. The left-to-right HMM state index is non-decreasing as the time increases.

i.e: $a_{ij} = P(Q_n = S_j | Q_{n-1} = S_i) = 0, \quad i > j.$

Non-emitting states can be used in an HMM model to indicate start and end states.

Figure 3.4 shows left-to-right and ergodic models using white circle as the emitting states and black circle as the non-emitting states. These states are used as the starting and ending point in the model which can be used in the concatenation between models.



(a) 3 state Left-to-right model (b) 3-state Ergodic model

Figure 3.4 HMM Modeling with emitting and non-emitting states

In training the HMM, we have samples of existing letters and their classes. Using the samples, an HMM is built for each letter in the set of letters. Thus, for each letter i, $1 \leq i \leq N$, where N is the total letters in the letter set, there is an HMM model $\lambda_i = \{A_i, B_i, \Pi_i\}$ being built.

For letter recognition, a new letter is given for recognition. The new letter is represented by the observation sequence, $O = O_1 O_2 \ldots O_T$. We need to decide which one of letter models λ_i, $1 \leq i \leq N$, best represent the observation sequence O. This is the evaluation problem of the HMM. First, we can compute $P(O|\lambda_i)$, which is the probability of the observation sequence O given the HMM model parameters for each of letters using the forward algorithm. Then the letter corresponding to the maximum probability, $P(O|\lambda_i)$, is chosen as the optimal answer. According to Bayesian classification theory, picking this letter minimizes the probability of error, therefore:

$$l_{optimal} = \arg\max_{l \in Nletters} P(O \mid \lambda_i) \qquad \text{(Eq. 3.31)}$$

3.4.2 Modeling Words

For handwriting recognition, a word can be modeled by an HMM if the lexicon is small in size. If the lexicon is large, normally the word HMM model is formed by concatenating letter HMMs since a word is made of a sequence of letters. In cursive writing, for words with the letters “i”, “j”, “x”, or “t”, where the writer adds the dots or crosses at the end of writing the word, these words are modeled with concatenating letter HMMs with letter-HMMs modeling these special characters, the “i” or “j” dot, the “t” cross, and the “x” cross, to the end of the HMM modeling these words. Some researchers, ignore these dots and crosses altogether.

Since these dots and crosses can be written in an arbitrary order, each of these words would have multiple word-HMMs representing each of them. The number of word-HMMs representing the same word can grow quite large as the number of “i”, “j”, “x”, or “t” letters increase. Researchers use various methods to represent these.

The simplest may be to represent these special letters by a single letter, such as the "backspace" character. For example, the HMM model of the word 'it' consists of four individual letter HMMs, each of which represents the letter "i", "t", "backspace", and "backspace", respectively. Using HMM with white and black states, the concatenation is formed at the black states. In this case, if a character model is to be removed, it can be modeled as allowing a transition of the initial state of a model letter to the final state of this same model letter.

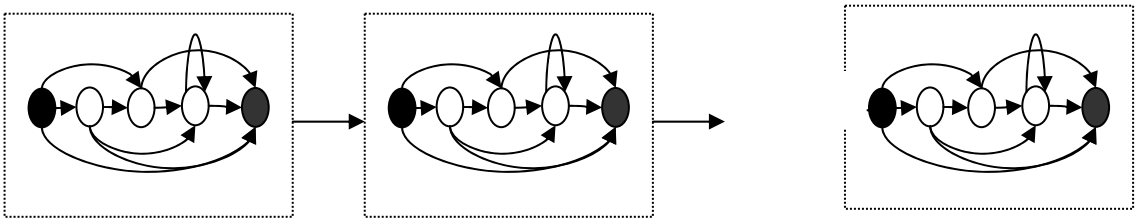


Figure 3.5 Concatenation of character HMM models to form a word model

A word HMM model therefore consists of a group of white emitting states and a group of black non-emitting states between initial state and final state of the model. In discrete density HMM, it is necessary to define a group of observation symbols and so, each white state has its own probability of emitting a symbol. The usual approach for defining the symbol is vector quantization method by using the k-means algorithm. In continuous density HMM, the probability is estimated on the space of observation possible. Among the popular one is Gaussian distribution.

For word recognition, word-HMMs built from the concatenation of letter HMMs are used to calculate the probability of the particular word given the observation sequence as described in section 1.4, using Bayes theorem. Words with the highest posterior probability are taken as the recognized word.

3.4.3 Modeling Sentences

Sentence model is formed from concatenation of word HMMs. In a sentence recognition system, the number of all possible words is normally limited to a particular recognition task. However, the numbers of sentences that can be composed with these words are very large. To model each sentence explicitly is simply computationally impossible. Fortunately, a probabilistic sentence network can be constructed to represent all of the possible sentences. Words in a sentence can also be assumed to satisfy the Markov condition, i.e: the word in a sentence is only dependent on the previous word, and not any other previous words.

The probability of a word given the previous word $P(Wn|Wn\text{-}1)$ is called the bigram probability. We can similarly have n-gram probabilities; the probability of a word given n-1 previous words. The bigram probabilities and the initial word probabilities, $P(W_0)$, together specify a bigram grammar for sentences. The probability of any sentence composed of a set of words, $W_0W_1...W_{n-1}W_n$, can be approximated with this bigram grammar:

$$P(W_0W_1...W_{n-1}W_n) \sim P(W_0)P(W_1|W_0),...P(W_{n-1}|W_{n-2})P(W_n|W_{n-1}) \quad \text{(Eq. 3.32)}$$

The bigram grammar can be estimated from a sentence data corpus. The sentences from the corpus are used to compute the bigram probabilities and initial word probabilities. To model all possible sentences made of the allowable words with HMMs, a bigram grammar can be constructed with all of these words using the sentence data corpus. Each node of a bigram grammar is actually a word HMM. These composite HMMs can represent all possible sentences made of all the allowable words.

In sentence recognition, new handwritten sentences in the form of the observation sequence $O = O_1O_2...O_T$ is given. A sentence corresponding to the observed sequence need to be obtained as the most probable sentence. Here, the HMM model parameters of each letter, $\lambda_i=\{A_i, B_i, \Pi_i\}$, are known, and the parameters of the bigram grammar are also known. Therefore, first, we calculate the optimal state sequence, $Q^* = Q_1^*Q_2^*... Q_T^*$ which corresponds to the observation

sequence using the Viterbi algorithm. Since the optimal state sequence is associated with a deterministic sequence of letters and words, this sequence of words is the desired result for the sentence.

On the extreme, a unique HMM can be created for each sentence, but it would be necessary to compute the probability $P(O|\lambda_i)$ for each sentence using forward algorithm. Since the number of possible sentences grows exponentially with the number of words, this method of utilizing the forward algorithm is computationally impractical. Therefore, it is necessary to use the Viterbi algorithm in order to solve the problem of sentence recognition.

3.5 Discriminative Training of HMM

Training of HMM using maximum likelihood estimation (MLE) approximation is known to be non-discriminative. This is due to the fact that only in-class examples are used in building the HMM models. To be discriminative, information from all examples of different classes need to be used in the model building. In MLE-based HMM training, the estimation process tries to optimize the modeling ability of the observation without having the measure of their classification ability. In real application, a good classifier needs to have the goal of better discrimination ability. There are some powerful HMM parameter estimation techniques and classifiers that use some form of discriminative information to achieve better classification. These other methods of training are the Maximum Mutual Information (MMI) training and the Minimum Classification Error Training (MCE) training. The motivation for MMI approach is based on the information theoretic concept and MCE approach is based on reducing classification error.

In the global picture of discriminative training, the focus is on directly modeling the boundary between classes. In this respect, discriminative training can be classified into either structure-bound or structure-free. In structure-bound methodology, discrimination ability is embedded within a preset classifier structure and the algorithm cannot be used independently of the structure. Examples of this are k-nearest neighbor classifiers and kernel-based methods such as Support Vector

Machines (SVM). Structure-free methodology uses an objective function that is independent of the system in which it is implemented and the same criterion can be used with various classifier structures.

MCE and MMI fall under Structure-free methodology. So are a few others like Minimum-Squared Error (MSE) and Cross-Entropy (CE). However, with the exception of MCE, the others are constrained on the architecture on which they are implemented. The CE criterion, which minimizes the cross-entropy between the target and the models, requires a probabilistic interpretation of the system's output. The MMI criterion which maximizes the mutual information between the data and their classes also requires a probabilistic interpretation of the system outputs. The MSE criterion, widely used in Neural Network-based learning, requires a target function and attempts to minimize the squared distance between the output of the system and the target.

In this section we review the two techniques for discriminative training of HMMs; the MMI and the MCE.

3.5.1 Maximum Mutual Information (MMI) training

The MMI criterion considers HMMs of all the classes simultaneously, during training. Parameters of the correct model are updated to enhance its contribution to the observations, while parameters of the alternative models are updated to reduce their contributions. This procedure gives a high discriminative ability to the system. In MMI training, we want to determine the components in the observation X that are most useful in distinguishing between the different classes in Y.

The mutual information I between X and Y is defined as the average amount of uncertainty about the knowledge (or the entropy) of X given the knowledge of Y (Cover, 1991) (Kullback, 1997). Mathematically this can be written as:

$$I(X;Y) = H(X) - H(X \mid Y) \qquad \text{(Eq. 3.33)}$$

The conditional entropy of X given Y is given by

$$H(X \mid Y) = \sum_{x,y} P(x,y)\ \ logP(x|y)\ \ = -E[logP(x|y)] \qquad \text{(Eq. 3.34)}$$

Putting handwriting recognition system in this mutual information framework, let *W* and *O* denote the random variables corresponding to the words and observation vectors. Similarly,the mutual information between *W* and *O* is given by:

$$I(W;O) = H(W) - H(W|O). \qquad \text{(Eq. 3.35)}$$

Thus, the uncertainty in the word given the sequence of observations is the conditional entropy of *W* given *O,* that is:

$$H(W|O) = H(W) - I(W;O). \qquad \text{(Eq. 3.36)}$$

We do not know *P(W,O)* in general and need to estimate it. The conditional entropy of the words given the observations *O* can be shown to satisfy the following inequality:

$$H_\lambda(W\,|O) \geq H(W\,|O) \qquad \text{(Eq. 3.37)}$$

where λ denotes a particular parametric estimate to the actual probability distribution. The equality holds only if $P_\lambda(W\,|O) = P(W\,|O)$. Thus by minimizing the conditional entropy $H(W|O)$ in (Eq. 3.36), we can get an estimate of the conditional distribution that minimizes the uncertainty of the data given the model. Minimizing $H(W|O)$ implies the maximization of $I(W;O)$, the mutual information, assuming a fixed $H(W)$. Thus this process is called maximizing the mutual information (MMI).

Using (Eq. 3.33) and (Eq. 3.34) we define an objective function, L_{MMI}, for the MMI estimation of the parameters, similar to the ML-based estimation of HMM parameters,

$$L_{MMI}(\lambda)\ \ = I_\lambda(W;O)\ \ =\ \ H_\lambda(W)\ -\ E[logP_\lambda\,(w|o)] \qquad \text{(Eq. 3.38)}$$

This objective function is the mutual information of the words given the observations under the parametric distribution. In this formulation we assume that

we have observations from a training set and that we can represent each observation as a composite HMM composed of a concatenation of letter HMMs representing the underlying observations.

Replacing the expectations by the sample averages and assuming the training data consists of R observations, we can write

$$L_{MMI}(\lambda) = -\frac{1}{R}\sum_{r=1}^{R}\log P_\lambda(w_r) \quad -\frac{1}{R}\sum_{r=1}^{R}\log\frac{P_\lambda(o_r \mid M_r)P_\lambda(w_r)}{P_\lambda(o_r)} \qquad \text{(Eq. 3.39)}$$

$$L_{MMI}(\lambda) = -\frac{1}{R}\sum_{r=1}^{R}\{\log P_\lambda(o_r \mid M_r) - \log P_\lambda(o_r)\}$$

In the above w_r is the word in the r[th] observation with a corresponding composite HMM model M_r. o_r are the set of observation vectors corresponding to the word. The first term in the above equation is the likelihood of the data given the model. Maximizing $L_{MMI}(\lambda)$ can be achieved by maximizing this likelihood, which is equivalent to ML estimation. However $L_{MMI}(\lambda)$ can also be maximized by simultaneously maximizing the first term in the right hand side of (Eq. 3.39) and minimizing the second term. The second term, the probability of the observation under a particular parameterization of the model, is what differentiates MMI from ML-based estimation. The probability of the observation can be defined in terms of the probability of generating all possible words.

$$P(o_r) = \sum_s P(o_r \mid M_{rs})P(M_{rs}) \qquad \text{(Eq. 3.40)}$$

where s represents any possible words and M_{rs} represents the composite observation model for a given word. Since the probability of the observation includes information comprised of both the correct and the incorrect hypothesis, this optimization process is more discriminative than the traditional ML-based estimation.

Details of practical implementation of MMI are discussed in (Bahl, 1992, Garcia-Salicetti, 1996, Bahl, 1986)

3.5.2 Minimum Classification Error (MCE) training

Both traditional ML and discriminative MMI techniques neither explicitly attempts to optimize the primary goal of a recognizer which is to maximize the rate of recognition, in other words, minimizing the error rate of recognition. MCE training technique directly minimizes errors. Because of that, as we mentioned earlier, MCE criterion is not limited to HMM parameter estimation and has been used to optimize several other types of classifiers including prototype based classifiers and neural networks.

The gist of MCE optimization is that we define a loss function in terms of the trainable parameters of the classifier that is proportional to the classification error. This loss function is then minimized using a suitable gradient-based technique. MCE training does not necessarily involve the estimation of probability distributions and hence no underlying probability distribution needs to be assumed. This circumvents a major drawback of ML estimation. MCE allows us to build classifiers that perform close to the Bayes error rate using the efficient method called Generalized Probabilistic Descent (GPD) which is based on Probabilistic Descent theorem by (Amari, 1967). The MCE/GPD paradigm was primarily applied to speech-related tasks, including acoustic modeling, word spotting, speaker recognition and adaptation, feature transformation, and feature extraction. Due to its success in speech recognition, recently, we have witnessed an increase in the number of MCE applications to handwriting recognition.(Biem, 2006)

The misclassification error measure in a classification problem can be defined in terms of discriminant functions of the k classes C_k, which is the word lexicon in the case of word recognition. We can choose a misclassification error such that it takes a value of zero for all correct classifications and non-zero values for misclassifications. This measure is not extremely useful because it does not provide a degree of separation between the correct and incorrect classes. In practice misclassification error measure with a gradual slope is preferred, such as the following:

$$d_k(O,\Phi) = -g_k(x,\Phi) + \left[\frac{1}{L-1}\sum_{j\neq k} g_j(O,\Phi)^{-\Psi}\right]^{-\frac{1}{\Psi}} \quad \text{(Eq. 3.41)}$$

where g_k is the discriminant function corresponding to the k^{th} class, Φ are parameters in the discriminant function, Ψ are parameters that controls the contribution of each misclassification towards the error metric and L is the number of classes in the classification problem. When Ψ is large, the most confusable class contributes the most to the summation.

In using the MCE framework in HMM parameter estimation, we start with the definition of the discriminant function in terms of the parameters of an HMM. In choosing the form of the discriminant function, the primary requirement is that the discriminant function can be used as a distance metric to compare classes. Normally, the likelihood of the class, C_j, in terms of the transition and observation probabilities is often used.

The likelihood is computed as the probability of all possible state sequences θ^p, for the given data. An expression for one particular state sequence can be written as follows:

$$f(O_1^T,\theta^p)\,|\,\Phi) = \prod_{t=1}^{T} a_{\theta_{t-1}^p \theta_t^p} . b_{\theta_t^p}(O_t) \quad \text{(Eq. 3.42)}$$

where a and b are the HMM transition and observation probabilities, respectively. Using the above definition of the likelihood, the discriminant function for the j^{th} class can be defined as follows:

$$g_j(O_1^T,\Phi) = \log\left[\sum_p [f(O_1^T,\theta^p)\,|\,\Phi)]^{\xi}\right]^{\frac{1}{\xi}} \quad \text{(Eq. 3.43)}$$

Note that when ξ is large, the most probable state sequence dominates the summation and the solution approach a Viterbi solution.

A loss function d can now be defined as the misclassification error measure. (Eq. 3.41) defines a commonly used loss function. This loss function is then minimized using gradient descent approaches similar to MMI estimation. From Amari's theorem, convergence to the local MCE optimum involves optimizing local loss functions. In general there are certain desirable properties for loss functions since GPD involves gradient computations. Near-binary functions are a desirable form for loss functions. Loss functions need to be first-order differentiable to apply GPD. A commonly used loss function that satisfies the above requirements is the sigmoid function:

$$l(d) = \frac{1}{1+e^{-\alpha d}} \qquad \text{(Eq. 3.44)}$$

where d is the misclassification error measure.

3.6 Discrete vs. Continuous Density HMM

A Comparison between continuous and discrete density HMM for cursive handwriting recognition has been done by (Rigoll, 1996). Discrete density HMM was shown to lead to better results than continuous Gaussian distribution HMM. This is generally not the case for HMM-based speech recognition systems. Although there are certain similarities between HMM-based speech recognition and handwriting recognition, different problems occur in both areas that it is not possible to handle the modeling problems for handwriting in exactly the same manner as for speech recognition. (Rigoll, 1996) performed systematic comparison between continuous and discrete density HMM for handwriting recognition using exactly the same databases for training and testing and conclude that discrete density HMM gives better recognition, especially for bigger database. Furthermore, discrete models allows for simpler feature extraction, data compression and speed advantage enabling towards a real time recognizer.

3.7 Hybrid of Neural Network and HMM

HMM deals with temporal aspects of handwriting efficiently because of the efficient training and decoding algorithms. However, many of the assumptions made in building and optimizing it, limit their generality. For HMM trained with MLE, besides the poor discrimination ability mentioned earlier, it also suffer from several drawbacks; (a) there need to be a priori choice of topology and initial probability distribution, (b) the first order Markov assumption for the state sequences, (c) uncorrelated input observation assumptions meaning that possible temporal correlation across features associated with the same HMM are totally disregarded.

To overcome the above problems, many researchers integrate neural network into the formalism of HMM. NN can approximate any kind of nonlinear discriminant function, flexible and do not need assumptions about the input distribution. The time sequences aspect that NN cannot handle is done by HMM. NN is used to estimate the probability of observation, which is the local posterior probabilities associated with each state in the HMM. These posterior probabilities are then turned into scaled likelihoods by dividing them by the estimated values of the class priors as observed in the training data. The scaled likelihoods are trained discriminatively using the ANN. During recognition, the scaled likelihoods are used in the Viterbi or forward computation to obtain an estimator of the global scaled likelihood.

In another form of hybrid NN/HMM, ANN can be trained according to the maximum a posteriori (MAP) criterion. The overall resulting training called recursive estimation and maximization of posterior probabilities (REMAP) becomes a form of EM training where posteriors are involved in the M step which is the NN training.

The NN can be one of the many possibilities available such as Time Delay Neural Network (TDNN), Space Displacement neural Network (SDNN), Convolutional Neural Network (CNN) etc.

3.8 Summary

In this chapter, we build the description of HMM from Markov process. The three problems in HMM are stated and shown how to solve them. Description of how HMM can be implemented in order to be used in on-line handwriting recognition is given at the letter, word and sentence level. In order to obtain a discriminative recognizer, methods of training HMM to be discriminative are given along with the method of using NN in the context of HMM to obtain an overall discriminative recognizer.

CHAPTER 4

SUPPORT VECTOR MACHINES

4.1 Introduction

In handwriting recognition systems based on Hidden Markov Model (HMM), handwriting is modeled by estimating a representation of the handwriting signal i.e. by estimating probability distributions of characters or words across the training data. The maximum likelihood (ML) based parameter estimation in an HMM tries to optimize the modeling ability without being able to measure their classification ability because only in-class data is used in the representation. (Riis, 1998)

Other useful HMM parameter estimation techniques discussed in chapter 3 use some form of discrimination information to achieve good classification. The discrimination information is in the form of an objective criterion that gives the probability of the data given the wrong model. Using the discriminative-based estimation, in-class (positive) examples and out-of-class (negative) examples are both used. This allows for simultaneously learning a good representation for in-class data while discriminating out-of-class data. Neural networks are discriminative classifiers because they learn the separating surfaces using both negative and positive examples. Classifiers that estimate decision surfaces directly have better performance than those that estimate a probability distribution across the training data.

In discriminative classifier point of view, handwriting recognition is a problem of supervised learning and classification. Handwriting data with known labels are

used to train a discriminative model. Then, for a new unseen handwriting, the model is used to predict the class label. A classifier can be a multiclass classifier such as in neural network or built from a number of basic two-class classifier into a multiclass classifier. The classifier model is said to generalize well if it can predict the correct classes well on a set of unseen test data.

Support Vector Machine (SVM) is a discriminative classifier that learns the decision surface through a process of discrimination and has good generalization characteristics. SVM have been proven to be a good classifier on many classical pattern recognition problems, among others; text categorization (Joachims, 1998), image recognition, image classification (Chapelle, 1998) (Chapelle, 1999), objects recognition, cancer classification (Chu, 2005), spam categorization (Drucker, 1999), face recognition, motion detection (Xu, 2005, Sidenbladh, 2004), face detection (Osuna, 1997); (Li, 2001), electricity fraud prediction (Ahmad, 2007), electricity load forecasting (Zhang, 2005), signature verification (Edson, 2005), time series prediction (Van Gestel, 2001), system identification (Zhang, 2004), web document classification (Lung, 2004), stock market forecasting (Huang, 2005), speech recognition (Joachims, 1999) (Ganapathiraju, 2004) and speaker verification (Wan, 2005).

SVM is quite a recent addition (1990s) to the various methods for classification. Its basic form implements a two-class classification method. It has been widely researched and used in recent years, for one, as an alternative to neural network. The main advantage of SVM, with respect to neural network, is that it provides a sound theoretical framework for taking into account not only the experimental data to design an optimal classifier, but also a structural behavior for allowing better generalization capability (Scholkopf, 1999). SVM generalization performance either matches or is significantly better than that of competing methods in most cases.

SVM's better generalization performance is based on the principle of structural risk minimization (SRM) (Vapnik, 1998). Its formulation approximates SRM principle by maximizing the margin of class separation. Thus, SVM classifier is also known as a large margin classifier. Basic SVM formulation is meant for linearly separable datasets. With a small modification, it can be used for non-linear datasets

by using kernel functions to indirectly map the nonlinear input space to a linear feature space where the maximum margin decision function is approximated (Burges, 1998). Outliers are handled through soft-margin SVM formulation. The general SVM formulation is non-linear soft-margin SVM in which linear and hard-margin (non-separable) problems are special cases.

SVM training involves approximating SRM by solving a convex quadratic programming problem with equality and inequality constraints. The final solution solves for nonzero parameters α in the formulation and extracts a subset of training data corresponding to the parameter. For training on small datasets, say less than 1000 samples, it can be solved reasonably fast and can be performed on a reasonably configured PC. For large datasets, solving the quadratic function requires large computing power and large memory for storage of the kernel matrix during computation. The memory requirement grows with the square of the size of training datasets.

A number of methods of SVM training have been developed over the years to improve on the memory requirement issue, speed up the training time and finding the best training model using appropriate kernel and the hyper parameters (Burges, 1998). In addition, since basic SVM can only handle two-class classification, to obtain multiclass classifier, at the minimum requires training of many two class classifiers and in classification, voting schemes are used for selecting the correct class (Weston, 1998) (Hastie, 1996) (Hsu, 2002). Method of modifying the two class SVM formulation into single multiclass formulation for solving simultaneous multiclass problem has been proposed but currently not widely implemented yet.

SVM have been made popular by the availability of stable implementation packages. There are a few implementation packages available publicly and have been popularly used as reported by many researchers. Among them are LIBSVM (Chang, 2001), SVMTorch (Collobert, 2001) and SVMLight (Joachims, 1999).

This chapter will introduce the theory behind SVM and demonstrate SVM implementations. We present the formulation of SVM in the next section, followed

by discussion on the different methods of implementing SVM for two class classification problems and expanding it to multiclass problems in section 4.3. The three publicly available SVM implementation packages mentioned earlier which we have tested are presented and compared in section 4.4.

4.2 Theoretical foundation

Vapnik has formulated the idea of support Vector Machines in the framework of Statistical Learning Theory (Vapnik, 1998). We first briefly discuss some basic ideas of the theory.

4.2.1 Statistical Learning Theory

In statistical learning theory (SLT), the problem of classification in supervised learning is formulated as follows (Vapnik, 1999):

> *We are given a set of ℓ training data and its class,* $\{(x_1, y_1),(x_2, y_{21}),...,(x_n, y_n)\}$ *in* $R^n \times R$ *sampled according to unknown joint probability distribution P(x,y) characterizing how the classes are spread in* $R^n \times R$.

We assume that the training data has been drawn randomly and independently based on the joint distribution. The goal of a learning machine is to learn the mapping $y = f(x)$. To learn the unknown mapping, we can perform either of the following:

(a) Estimate a function that is "close" to the joint distribution under an appropriate metric.
(b) Learn an optimal predictor or classifier of the systems output.

In the former case, it is not sufficient for us to estimate a good predictor of the output. The goal is to estimate, the joint probability distribution. However, for data classification, we can actually pursue the goal of learning an optimal predictor or

classifier. Learning is then a process of choosing a function from a set of functions defined by the construction of the learning machine. For a gradient-based neural network classifier, the network structure is predefined, leading us to choose from only a finite set of functions. This is accomplished by finding the weights of the connections of the predefined network. The optimal network for the classifier is chosen based on some optimality criterion that measures the quality or performance of the learning machine. SLT allows us to learn the optimal classifier by minimizing the structural risk.

4.2.2 Structural Risk Minimization

To measure the performance of the classifier, a loss function *L(y,f(x))* is defined as follows:

$$L(y,f(x)) = \begin{cases} 0 \text{ if } y = f(x) \\ 1 \text{ if } y \neq f(x) \end{cases} \quad \text{(Eq. 4.1)}$$

i.e. *L(y,f(x))* is zero if *f* classifies x correctly, one otherwise.

On average, how *f* performs can be described by the Risk functional

$$R(f) = \int L(y,f(x))\, dP(x,y) \quad \text{(Eq. 4.2)}$$

Since *P(x,y)* in unknown, an estimate of the risk (the empirical risk) can be obtained by induction using principle of empirical risk minimization (ERM) over a set of possible functions as follows:

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)) \quad \text{(Eq. 4.3)}$$

ERM principle states that given the training set and a set of possible classifiers in the hypothesis space *F*, we should choose $f \subset F$ that minimizes $R_{emp}(f)$. ERM is one

of the most commonly used optimization procedures in machine learning. It is computationally simpler than attempting to minimize the actual risk as defined in (Eq. 4.2). ERM circumvents the need for the estimation of the joint probability density function. In many cases, ERM provides a good quality learning machine. A variety of loss functions can be used for the optimization process. One such example is,

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l}|y_i - f(x_i)| \qquad \text{(Eq. 4.4)}$$

where y is the output of the classifier and x is the input vector. This form of a loss function is common in learning binary classifiers. For example, to estimate the parameters of a multi-layered perceptron using the back-propagation algorithm, a loss function representing the squared error is used.

However, ERM does not necessarily produce a good classifier, which generalizes well to unseen data due to overfitting phenomena. $R_{emp}(f)$ is a poor, over-optimistic approximation of $R(f)$, the true risk. There could be several configurations of the learning machine, which give us the same empirical risk (zero, in the case of binary classifiers). How then can we choose the best configuration?

The normal practice to get a more realistic estimate of generalization error, as in neural network is to divide the available data into training and test set. Training set is used to find a classifier with minimal empirical error (optimize the weight of an MLP neural networks) while the test set is used to find the generalization error (error rate on the test set).

If we have different sets of classifier hypothesis space F_1, F_2 ... *e.g.* MLP neural networks with different topologies, we can select a classifier from each hypothesis space (each topology) with minimal $R_{emp}(f)$ and choose the final classifier with minimal generalization error. Of course, to do that requires designing and training potentially large number of individual classifiers.

Using SLT, we do not need to do that. Generalization error can be directly minimized by minimizing an upper bound of the risk functional *R(f)*. Let us analyze the relationship between the actual risk and the empirical risk. The bound given in (Eq. 4.5) holds for any distribution *P(x,y)* with probability of at least 1- η

$$R(f) \le R_{emp}(f) + \phi(\frac{h}{l}, \frac{\log(\eta)}{l}) \qquad \text{(Eq. 4.5)}$$

where the parameter h denotes the so called VC (Vapnik-Chervonenkis) dimension. ϕ is the confidence term defined by Vapnik as :

$$\phi(\frac{h}{l}, \frac{\log(\eta)}{l}) = \sqrt{\frac{h(\log\frac{2l}{h}+1) - \log(\frac{\eta}{4})}{l}} \qquad \text{(Eq. 4.6)}$$

ERM is not sufficient to find good classifier because even with small $R_{emp}(f)$, when h is large compared to l, ϕ will be large, so $R(f)$ will also be large, ie: not optimal. We actually need to minimize $R_{emp}(f)$ and the confidence term ϕ at the same time, a process which is called structural risk minimization (SRM). By SRM, we do not need the test set for model selection anymore. Taking different sets of classifiers F_1, F_2 ... with known h_1, h_2 ... we can select f from one of the set with minimal $R_{emp}(f)$, compute $\phi(\frac{h}{l}, \frac{log(\eta)}{l})$ and choose a classifier with minimal $R(f)$. No more evaluation on test set is needed, at least in theory. However, we still have to train potentially very large number of individual classifiers. To avoid this, we want to make h tunable (ie: to cascade a potential classifier F_i with VC dimension = h and choose an optimal f from an optimal $\boldsymbol{F_i}$ in a single optimization step. This is done in large margin classification of which SVM is one.

4.3 SVM Formulation

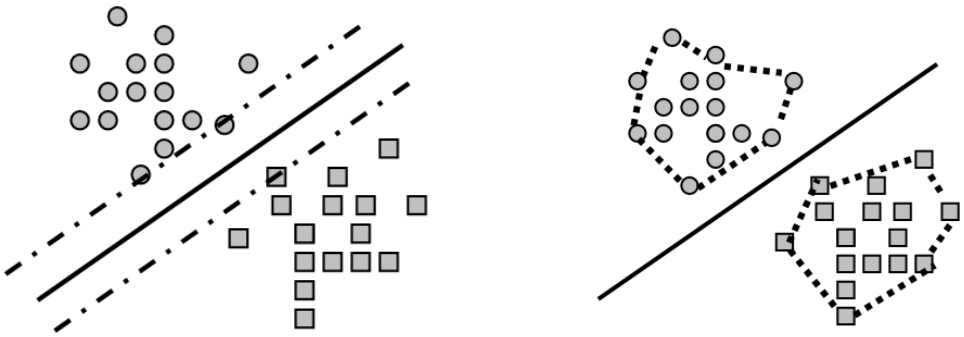
SVM is realized from the above SLT framework. The simplest formulation of SVM is linear, where the decision hyper plane lies in the space of the input data x. In this case the hypothesis space is a subset of all hyper planes of the form:

$f(x) = w \cdot x + b$. SVM finds an optimal hyper plane as the solution to the learning problem which is geometrically the furthest from all classes since that will generalize best for future unseen data.

There are two ways of finding the optimal decision hyper plane. The first is by maximizing the margin between two supporting planes as shown in Figure 4.1(a). The second is by finding a plane that bisects the two closest points of the two convex hulls defined by the set of points of each class, as shown in Figure 4.1(b). Both methods will produce the same optimal decision plane and the same set of points that support the solution (the points on the two parallel supporting planes in Figure 4.1(a) or the closest points on the two convex hulls in Figure 4.1(b). These are called the support vectors.

4.3.1 Linearly Separable Case

Let's consider SVM formulation for linearly separable case using the method of maximizing margin as outlined in Figure 4.1(a). For a set of l linearly separable data $\{(x_1, y_1),(x_1, y_1),...(x_1, y_1)\}$ where $x_i \in R^d$ and $y_i \in \{\pm 1\}$ we would like to learn a linear separating hyper plane classifier $f(x) = w.x + b$ that has the maximum separating margin with respect to the two classes where w is the normal of the hyperplane.



(a) maximal margin between two supporting planes

(b) Optimal plane bisects closest points in convex hulls

Figure 4.1 Finding the optimal decision hyperplane

We specifically want to find the hyperplane: $H: y = w.x + b = 0$ and two hyper planes parallel to it and with equal distances to it,

$H_1: y = w.x+b = +1$ and
$H_2: y = w.x + b = -1$

with the condition that there are no data points between H_1 and H_2, and the distance or margin M between H_1 and H_2 is maximized. Figure 4.2 show the hyper planes in the case of input data x with two dimensions.

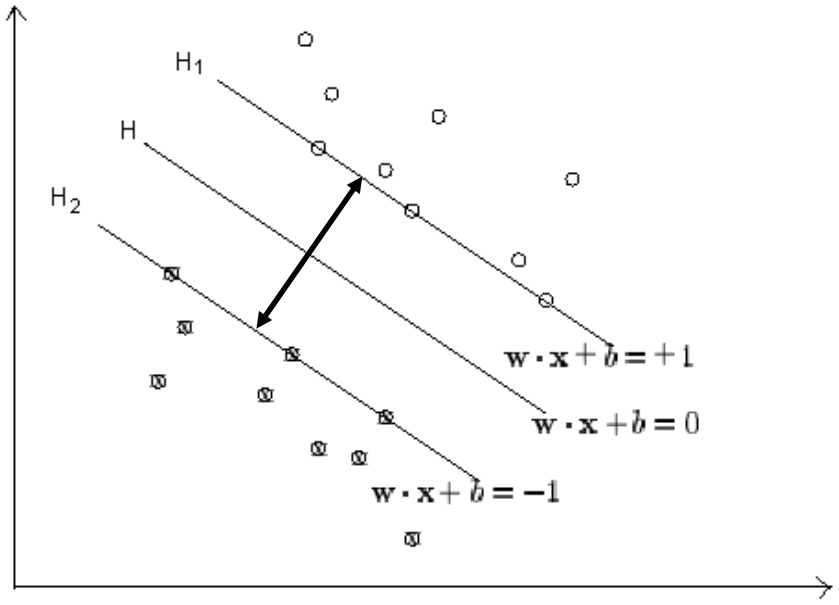


Figure 4.2 Maximal Margin hyperplanes for two dimension examples

For any H, H_1 and H_2, we can always normalize the coefficients vector w so that:

H_1 be $y = w.x + b = +1$, and
H_2 be y = w.x + b = -1.

We want to maximize the distance between H_1 and H_2. Therefore, there will be some positive examples on H_1 and some negative examples on H_2. These are the support

vectors. The distance between H_1 to H is $\frac{|wx+b|}{\|w\|} = \frac{1}{\|w\|}$ and thus between H_1 and H_2 is $\frac{2}{\|w\|}$. Therefore, to maximize the margin, we need to minimize $\|w\| = w^T w$ with the condition that no data points lies between H_1 and H_2. This is satisfied when:

$$w.x + b \geq +1 \quad \text{for } y_i = +1,$$
$$w.x + b \geq -1 \quad \text{for } y_i = -1.$$

Combining the two conditions, we have: $y_i(w.x + b) \geq 1$

For simplicity, the problem can be formulated as:

$$\min_{w,b} \frac{1}{2} w^T w, \qquad \text{(Eq. 4.7)}$$

subject to $y_i(w.x + b) \geq 1$.

This can be solved by introducing Lagrange multipliers $\alpha_1, \alpha_2, \ldots, \alpha_l \geq 0$, for every training data (Klein, 2000). See appendix B for a discussion on Lagrange multipliers.

Thus, we have the following Lagrangian:

$$L(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{l} \alpha_i y_i (wx_i + b) + \sum_{i=1}^{l} \alpha_i \qquad \text{(Eq. 4.8)}$$

This is called the primal formulation of the optimization problem and we often denote it as L_P. The first term on the RHS, defined as half the square of the norm, is called the objective function and the other two terms are the optimization constraints. This is a convex quadratic problem (because the objective function itself is convex). We have to maximize L_P with respect to α, subject to the constraint that the gradient of L_P with respect to the primal variables w and b should be 0:

i.e: $\frac{\partial L_P}{\partial w} = 0$ and $\frac{\partial L_P}{\partial b} = 0$ and that $\alpha \geq 0$.

Finding the gradient and solving for 0, we then have:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad \text{(Eq. 4.9)}$$

and

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \qquad \text{(Eq. 4.10)}$$

Substituting them into L_P, we have the Lagrangian dual L_D where:

$$L_D = \sum_{i=1}^{l} \alpha_i y_i x_i . \sum_j \alpha_j y_j x_j - \sum_{i,} \alpha_i y_i (x_i . \sum_j \alpha_j y_j x_j + b) + \sum_i \alpha_i$$

$$= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \qquad \text{(Eq. 4.11)}$$

using (Eq. 4.12). Observe that the primal variables *w* and *b* are eliminated.

Solving for α_i, using L_D constitute SVM learning. In order to obtain the value of *w* we substitute α_i into the formula $w = \sum_{i=1}^{l} \alpha_i y_i x_i$. The value of b can be averaged from the values of $y - wx$ for each x in the training set, after *w* is obtained. Thus, we obtained the decision function as:

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i \, y_i (x_i . x) + b) \qquad \text{(Eq. 4.12)}$$

where the sign *(sgn)* is used to classify examples as either in-class or out-of-class.

In other words the above equation defines the SVM classifier. Observe that the classifier is defined in terms of the training examples. However, all training examples do not contribute to the definition of the classifier. The training examples with non-zero multipliers, the Support Vectors, alone define the classifier. The dataset size can also define how complex the classifier needs to be. In simple classification problems the number of support vectors is normally small; and vice versa. The complexity of the classifier scales linearly with the number of support

vectors, because since there are M dot products involved in the definition of the classifier, where M is the number of support vectors.

4.3.2 Optimality Condition

In the above Eq. 4.11, the optimization of L_D is subject to the positivity of α_i and the constraints in Eq. 4.10. Because here we have the optimization of a convex function constrained by concave functions, Karush Kuhn-Tucker (KKT) theorem (Kuhn, 1951) applies. The theorem guarantees that non-negative Lagrange multipliers exist. See Appendix 2 for Discussion on Lagrange and KKT.

An issue of importance in the optimization using Lagrange multipliers is on the existence of an optimum. In addition, if an optimal point exists, we need to know if it is guaranteed that there exists a single optimal point. The answer to this question lies in the KKT theorem that guarantees the existence of a solution and prescribes a set of necessary and sufficient conditions. The KKT theorem has been widely used in optimization problems involving convex objective functions. For SVM problem of finding the optimal hyperplane, the KKT conditions are used in formulating the constraints. The positivity constraint on the Lagrange multipliers as mentioned earlier is one such example.

In the SVM optimization process, using the third KKT conditions with the Eq. 4.7, and the condition $y_i(w.x+b) \geq 1$, we get

$$\alpha_i(y_i(x_i+b)-1)=0 \qquad \text{(Eq. 4.13)}$$

The above equation implies that α_i is non zero only for examples that satisfy,

$$y_i(x_i+b)=1 \qquad \text{(Eq. 4.14)}$$

which are the support vectors. Eq. 4.13 also helps the optimization process in identifying examples that violate the KKT conditions, which will not be part of the support vectors. Identifying such examples helps in speeding up optimization process. It also allows the handling of large datasets efficiently.

4.3.3 Linear Soft Margin and Non-Linear SVM

Due to nonlinearities or noise, real world data is usually not linearly separable. In the case of imperfectly separable input space, where noise in the input data is considered, there is no enforcement that there be no data points between the planes H_1 and H_2 mentioned in the previous section, but rather penalty C is enforced if data points cross the boundaries. So, the the problem can be formulated as:

$$\min_{w,\xi,b} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \qquad \text{(Eq. 4.15)}$$

where C is the penalty term, subject to the condition $y_i (w.x + b) \geq 1 - \xi_i$.

Using similar formulation as in linear case, we obtained the same dual Lagrangian but with a different constraint for α_i, which is bounded above by C (ie: $0 < \alpha_i < C$). For non-linearly separable input, they can be mapped to higher dimensional feature space as mentioned earlier. If the mapping function is $\Phi(.)$, we just solve:

$$\max L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(x_i) \Phi(x_j) \qquad \text{(Eq. 4.16)}$$

Generally, if the dot product $\Phi(x_i).\Phi(x_j)$ is equivalent to a kernel $K(x_i, x_j)$, the mapping need not be done explicitly. Thus, equation above can be replaced by:

$$max L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad \text{(Eq. 4.17)}$$

Using the kernel in input space is equivalent to performing the map into feature space and applying dot product in that space. Many kernels can be used in that way as long as they satisfy Mercer's condition. Table 4.1 gives a number of commonly used kernels.

Table 4.1 Commonly used Kernels for SVM

Kernel type	Equation
Linear kernel	$K(x, y) = x.y$
Polynomial kernel	$K(x, y) = (x.y + 1)^d$
Radial basis function (Gaussian) kernel	$K(x\ y) = e^{\frac{-\lvert x-y\rvert^2}{2\sigma^2}}$
Hyperbolic tangent kernel	$K(x, y) = tanh(ax.y - b)$

Beside the above kernels, user defined kernels can also be used as long as they satisfy Mercer's condition (Joachims, 1999). (Burges, 1998) also gives a good description of Mercer's condition.

4.3.4 Variations of the SVM Objective Function.

There are many possibilities of penalizing outliers. The most common one is by using l_1 norm as in equation (Eq. 4.1). Another penalizing term would be squared quadratic norm l_2. Using l_2 norm, the problem can be formulated as:

$$\min_{w,\xi,b} \frac{1}{2} w^T w + \frac{1}{2} C \sum_{i=1}^{l} \xi_i^2 \qquad \text{(Eq. 4.18)}$$

subject to the condition $y_i(w.x + b) \geq 1 - \xi_i$, $\alpha \geq 0$ where C is the penalty term, and the dual optimization problem will have $\frac{1}{C}$ added to the every element of the kernel diagonal matrix. This is easier to solve than quadratic program with the additional constraint on α. But solutions for SVM formulation with l_2 norm are often less sparse than for the l_1 norm resulting in more support vectors being used in the separating hyper plane. Sparseness can be enforced directly by using l_1 norm for the normal w as well. This leads to the so-called linear programming (LP) SVM where the following linear program must be solved:

$$\min_{w,\xi,b} \sum_{i=1}^{l} w_i + C \sum_{i=1}^{l} \xi_i \qquad \text{(Eq. 4.19)}$$

subject to the condition $y_i(w.x+b) \geq 1-\xi_i$, $\alpha \geq 0$ and $\xi \geq 0$. LP SVM works well for very large linear data set and by the use of kernel; it can be turned to work for non-linear dataset quite well.

Scholkopf (Scholkopf, 1999) proposed another SV algorithm called v-svm which uses v parameter for controlling the number of support vectors. The formulation is as follows:

$$\min_{w,\xi,b} \sum_{i=1}^{l} w_i - v\rho + \frac{1}{l} \sum_{i=1}^{l} \xi_i^2 \qquad \text{(Eq. 4.20)}$$

subject to the condition

$$y_i(w.x+b) \geq \rho - \xi_i, \quad 0 \leq v \leq 1 \text{ and } \xi_i \geq 0, \rho \geq 0 .$$

Most SVM formulations however, are based on the 'classical' formulation with l_1 and l_2 norm on slack variables and l_2 norm for w. Our implementation discussion in the next section is based on that formulation.

4.4 SVM Implementations

Implementing SVM training involves the following steps:

- Select the parameter C (representing the tradeoff between minimizing the training error and margin maximization), kernel function and any kernel parameters.

- Solve the dual QP (Eq. 3.10) or alternative problem formulation using appropriate QP or LP algorithm to obtain the support vectors.

- Calculate threshold b using the support vectors.

SVM classification can then be done using the formula:

$$f(x) = \text{sgn}(\sum_{i=1}^{N} \alpha_i \, y_i K(x_i . x) + b) \, .$$

One of the problem in SVM training is to select the parameter values C and the kernel parameters. This is known as model selection. Kernel parameters are referred to as hyper parameters. Choosing hyper parameters involves minimizing an estimate of generalization error or some related performance measures. Among those estimates are k-fold cross-validation and leave-one-out (LOO) estimates which is the extreme of k-fold cross validation. In k-fold cross validation, training data is randomly split into k mutually exclusive subsets or folds of approximately equal size. SVM decision function is obtained using k-1 of the subsets and tested on the subset left out. This is repeated k times. Averaging over the k trials gives estimate of the expected generalization error. Other recent model selection strategies are based on some bound, which can be determined by a quantity, which is not obtained, using retraining with data points left out as in cross-validation or LOO.

4.4.1 QP Optimization

Typically, solving the QP or LP problem is a well-studied field of mathematical programming. The QP problem is solved by moving between the primal formulation and its dual formulation. However, existing general-purpose QP algorithms can only handle small sized problems. They are not feasible if the kernel matrix is large (do not fit memory of the running computer) due to large number of training inputs.

There are *l* free parameters in an SVM trained with *l* examples. The parameters are the α_i's. To find these parameters, the quadratic programming problem is solved subject to the constraints. Conceptually, the problem is to find a minimum of a bowl-shaped objective function. The QP iteration has definite termination conditions, the Karush-Kuhn-Tucker conditions mentioned in section 4.2.4, that describe the set of α_i that are the minima. In earlier SVM implementations, a QP optimizer routine is

normally used but it is slow and does not work well on large problems. Alternative optimization techniques that can be used are;

(a) techniques in which kernel components are evaluated and discarded during learning,
(b) decomposition method in which an evolving subset of data is used.
(c) new optimization approaches that specifically exploit the structure of the SVM problem.

Kernel Adatron (Friess, 1998) is one of the method which follows (a). It sequentially updates the alphas. It is very easy to implement but is not as fast as using QP routines. Technique (b) involves chunking and decomposition. Rather than sequentially updating the α_i, the α_i's are updated in parallel but using only a small subset or working set of data at each stage. There are many formal algorithms developed using chunking and decomposition for solving the optimization problem of support vector machines Among them, are Chunking (Osuna, 1997), Sequential Minimal Optimization (SMO) (Platt, 1998b), (Platt, 1999b) and SVMlight (Joachims, 1999).

In chunking, some QP optimization algorithm is used to optimize the dual QP on an initial arbitrary subset of the data. The support vectors found are retained and all other data points with α_i equal zero are discarded. A new working set of data is then derived from these support vectors and additional data points that maximally violate the constraints. This chunking process is then iterated until the margin is maximized. The chunking algorithm starts with an arbitrary subset (chunk of data, working set) which can fit in the memory and solves the optimization problem on it by the general optimizer. Support vectors (SVs) remain in the chunk while other points are discarded and replaced by a new working set with gross violations of KKT (Karush-Kuhn-Tucker) conditions (Osuna, 1996). The rationale of this operation is that only support vectors contribute to the final form of a decision function. In addition, the chunking algorithm is based on the sparsity of SVM's solution. That is, support vectors actually take up a small fraction of the whole data set. However, the problems with chunking is that there may be many more active candidate support vectors during the optimization process than the final ones so that their size can go

beyond the chunking space. The method of selecting a new working set by evaluating KKT conditions without efficient kernel caching may require high computation.

Decomposition, use a fixed-sized subset of data – the working set with the remainder kept fixed. A much smaller QP or LP is solved for each working set. Thus, many small sub problems are solved instead of one massive one. The limiting case of decomposition is in the sequential Minimal Optimization (SMO) by Platt (Platt, 1998a). Platt decomposes the overall QP problem into size of two, ie: two α_i are jointly optimize analytically at each iteration. This eliminates the need for a QP solver for the sub problem which is a plus point. Furthermore, an analytical solution for a two-point optimization problem can be given explicitly. The method consists of a heuristic step for finding the best parameters to optimize and use an analytic expression to ensure the dual objective function increases monotonically. Several heuristics have been suggested to select the working set. The original SMO, was then improved by Keerthi and Shevade (Shevade, 2000).

Keerthi further enhance the performance of SMO by pointing out the inefficiency of updating one-thresholded parameters in Platt's algorithm and replacing it with two-thresholded parameters. The important contribution of Keerthi et al.'s modification is that the pair of patterns chosen for optimization is theoretically determined by two-thresholded parameters and the optimization on this subset leads to a considerable advancement in the objective function. In practice, when the size of a data set grows bigger, the problem of determining the optimal pair at a low cost still exists.

SVMlight (Joachims, 1999) is a general decomposition algorithm, where a good working set is selected by finding the steepest feasible direction of descent with q nonzero elements. The q variables that correspond to these elements compose the working set. When q is set equal to 2, Chang and Lin (Chang, 2001) pointed out that the selected working set corresponds to the optimal pair in Keerthi et al.'s modification of SMO. SVMlight caches q rows of kernel matrix (row caching) to avoid kernel reevaluations and LRU (Least Recently Used) is applied to update the

rows in the cache. However, when the size of the training set is very large, the number of cached rows becomes small due to the limited memory. As a result, the number of active variables is not large enough to achieve a fast optimization.

The final approach is to directly attack the SVM problem from an optimization perspective and create algorithms that explicitly exploit the structure of the problem. These involve reformulation of the base SVM problem. The reformulation has been proved as effective as the original SVM in many cases. Keerthi (Keerthi, 1999b) proposed the nearest point algorithm (NPA) based on the idea described in section 4.2., which is to find the two closest points in the convex hulls. This method however is not very popular.

For solving SVM's learning problem on a very large data set, many researchers propose different methods. Collobert et al. (Collobert, 2001) proposed a parallel mixture of SVMs. The model first trained many SVMs on small subsets and then combined their outputs using a gater such as linear hyper plane or multilayer perceptron. However, there are problems with that; first, is to determine the optimal number of local SVMs, second is that generalization performance in not well achieved.

Another approach is to apply the Bayesian committee machine (BCM) (Tresp, 2000) to the support vector machine resulting in Bayesian committee support vector machine (BC-SVM) (Schwaighofer, 2001). In the BCM, the data set is divided into M subsets of the same size and M models are derived from the individual sets. The predictions of the individual models are combined using a weight scheme, which is derived from a Bayesian perspective in the context of Gaussian process regression. That is, the weight for each individual model is the inverse covariance of its prediction. A good approximation requires that M subsets be pair wisely independent. Although the Bayesian committee support vector machine performs better than uniform mixture of individual SVMs on subsets, it has a slightly higher error rate than the full SVM on some data sets (Schwaighofer, 2001), (Kuhn, 2006).

4.4.2 Multiclass SVM Implementation

As opposed to neural network, SVM is a two class classifier. Multiclass SVM, is formulated in either one of two ways; first, by combining binary classifiers or second, by modifying 2 class SVM to incorporate multiclass learning (Hsu, 2002). In the first way, multiple 2-class classifiers such as 1 against 1 or 1 against the rest are constructed and during classification, each classifier outputs are combined in some way into multiclass classifiers.

For 1 against 1 method, in a *k* class problem, *k(k-1)/2* classifiers are constructed and for classification, voting method or directed acyclic graph (DAG) can be used to combine the two class classifiers. Using DAG, each internal node is a 1 against 1 classifier and all leaf nodes are the classes. For recognition, the graph is traversed through from the root and arriving at the leaf with the correct classification. In 1 vs all method, k classifiers are constructed. For recognition, classifier with the highest output is chosen as the correct class.

For the second way, multiclass classifier is constructed by solving one complex optimization problem involving large number of free parameters. This all-together method have been proposed by Weston and Vapnik. (Weston, 1998). 1 against 1 method with DAG is widely used since it is less complex, easy to construct and faster to train.

4.4.3 SVM Posterior Probability Output

In hybrid connectionist systems which are used in speech and handwriting recognition so far, neural network is the main technology used to estimate posterior probabilities of emission for each observation and HMM is used to model temporal evolution. When SVM is used in such a hybrid system in place of ANN, several issues arise due to SVM optimization method and the output that it gives. Basic SVM provides binary decision values (which are the two classes) and multiclass SVM gives m-ary decision values (which are the m classes). Most applications require posterior probability values that capture uncertainty in classification. An

example, for the 2-class SVM is P(y = 1 | x), the probability that the input belongs to one of the particular class.

The first issue is how to estimate the output of the SVM into posterior probability. There is a lack of clear relationship between SVM output and the posterior class probability. (Ganapathiraju, 2002) in his thesis discussed ways of converting the posterior to a probability, such as fitting Gaussian and histogram approaches. However, these methods are not Bayesian in nature in that they do not account for the variability in the estimates of the SVM parameters. Ignoring this variability in the estimates often results in overly confident predictions by the classifiers on the test set.

(Allwein, 2000) and (Kwok, 1999) used moderated SVM outputs as estimates of the posterior probabilities. Kwok extend the use of moderated outputs to SVM by making use of a relationship between SVM and the evidence framework. The moderated output is in line with the Bayesian idea that the posterior weight distribution should be taken into account in prediction. It also alleviates the usual tendency of assigning overly high confidence to the estimated class memberships of the test patterns. Normally, unmoderated probability estimates based on maximum likelihood (ML) fitting can be fairly used as a trade-off between computational complexity and error performance. Mapping the output distances to posteriors is done using a sigmoid distribution:

$$p(y = 1 \mid f) = \frac{1}{1 + e^{Af + B}} \qquad \text{(Eq. 4.21)}$$

(Platt, 1999a) first trained an SVM and then train the parameters of the sigmoid function above to map the SVM output to probabilities. The values of A and B are found by minimizing the negative log likelihood of the training data. For example given a training set, a subset of *l* training data (N_+ of them with class y = +1, and N_- of them with class y = -1) can be used to solve the following maximum likelihood problem:

$$\min_{z=(A,B)} F(z) \qquad \text{(Eq. 4.22)}$$

where

$$F(z) = -\sum_{i=1}^{t} (t_i \log(p_i) + (1 - t_i) \log(1 - p_i)),$$

$$p_i = p(y = 1 \mid f) = \frac{1}{1 + e^{Af+B}}, \quad f_i = f(x_i) \text{ and}$$

$$t_i = \begin{cases} \dfrac{N_+ + 1}{N_+ + 2} & \text{if } y_i = 1 \\ \dfrac{1}{N_- + 2} & \text{if } y_i = -1 \end{cases} \qquad i = 1,2,...,l$$

4.5 SVM Implementation Packages

There are many publicly available SVM packages made available by researchers. They are either Matlab based toolbox modules or libraries implemented in C or C++ code. This section introduces three most widely used SVM packages: SVMTorch, SVMlight and LIBSVM. In section 6.3, we compared each of the package for their suitability, in order to adapt one of them in our hybrid SVM/HMM handwriting recognizer.

4.5.1 SVMTorch

SVMTorch was developed by Ronan Collobert (Collobert, 2001). It follows the same principle as the one used in the other two packages and works both for classification and regression. It is tailored for large-scale problems (with > 20000 examples, and input dimensions >100). The code is written in C++ and it is optimized for a working set of size 2. Working set selection follows the idea proposed by Joachim for his SVMlight. An internal cache for keeping part of the

kernel matrix in memory enables the program to solve large problems without the need to keep quadratic resources in memory and without the need to recompute every kernel evaluation.

Shrinking is an option where some variables whose values have been equal to the bounds 0 or C for a long time and probably will not change are removed. Input can be provided in either sparse, non sparse or binary file format. It can handle multiclass as well. The multiclass implementation in SVMTorch is based on one against all and selecting the highest score among all classifier outputs for classification. The package runs on Solaris, Linux and Windows platforms. The algorithm has been proved to converge for any working set size but without shrinking. SVMTorch consists of a learning module (svmtorch) and a classification module (svmtest).

4.5.2 SVMLight

SVMlight by Joachim (Joachims, 1999) implements SVM for classification, regression and ranking problems in C language. It uses a fast optimization algorithm based on decomposition. Working set selection is based on steepest feasible descent. Shrinking and caching of kernel evaluations are used. It can handle large data set (>100,000 training examples). It supports standard as well as user defined kernel functions. Input can be either dense or sparse. The optimization algorithms used in SVMlight has scalable memory requirements and can handle large problems efficiently. The package also provides methods for assessing the generalization performance efficiently in the form of the XiAlpha-estimates and leave-one-out (LOO) testing of the error rate, the precision and the recall.

Ranking functions (learning a function from preference examples, so that it orders a new set of objects as accurately as possible) and training of large-scale transductive SVMs are also provided in the tool package. SVMlight consists of a learning module (svm_learn) and a classification module (svm_classify).

4.5.3 LIBSVM

LIBSVM (Chang, 2001) is an integrated package for support vector classification; C-SVC and nu-SVC, regression; epsilon-SVR and nu-SVR and distribution estimation or one-class SVM. It also supports multi-class classification. The basic algorithm is a simplification of both SMO by Platt and SVMLight by Joachims. It is also a simplification of the modification of SMO by Keerthi et al . (Keerthi, 1999a).

LIBSVM was developed as a general-purpose SVM tool. It provides a simple interface to link the tool with the adapter of the tool. It is flexible in that it uses a number of different SVM formulations, user selectable via command line arguments. It also allows for automatic cross validation functionality for model selection. A contour of cross validation accuracy can be generated. In cases of unbalanced data the SVM can be configured so that it will function as weighted SVM. LIBSVM consists of a learning module (svmtrain) and a classification module (svmpredict).

4.6 Summary

In this chapter, we describe SVM as a discriminative classifier that learns the decision surface through a process of discrimination and has good generalization characteristics. SVM uses both training data and structural behavior to achieve better generalization capability than that of competing methods. We described the principle of structural risk minimization (SRM) which is the basis for SVM, a large margin classifier. We also discussed SRM and its relationship to ERM. The control over the generalization offered by SRM is what makes an SVM a very powerful machine learning technique.

The design and construction of maximum margin hyper planes which form the core of SVM estimation was discussed. SVM formulation for linearly separable datasets and non-linear datasets using kernel functions are laid out. For non separable case, soft-margin SVM formulation was used. Several issues related to

SVM training were discussed; in particular using the convex quadratic programming problem with equality and inequality constraints, ways of improving memory issues, speed of the training and selecting the best training model using appropriate kernel hyper parameters. Finally we discussed SVM implementation packages.

CHAPTER 5

HYBRID SVM/HMM HANDWRITING RECOGNITION SYSTEM

5.1 Introduction

Chapter 3 and 4 has prepared us to describe the online handwriting recognition system used in this work. Our final aim is to build an INSEG based word recognizer using hybrid of support vector machine (SVM) and hidden markov model (HMM). In the final setting, the SVM which is used to recognize characters or sub characters needs to be trained with characters which are segmented from the word database. SVM trained using separate isolated characters is not suitable since an isolated character is different from the same character which is written as part of a word.

In building the character recognizer, we started off by building a character recognizer trained using the isolated characters from IRONOFF and UNIPEN databases. We used this to compare recognition results of the SVM based character recognizer and other methods. In this chapter, we first describe the character recognizer based on SVM. Then we describe the word recognition system that makes use of the SVM based character recognizer in a hybrid SVM and HMM situation. The training of the hybrid system is separately at the character and word level. In a way it is based on the original approach of hybrid Neural network (NN) and HMM by (Gilloux, 1995) but it is also influenced by work of (Tay, 2003) for offline handwriting.

5.2 Overview of the SVM based Character Recognizer

The initial character recognition system makes use of isolated characters in its training. We trained and tested the character recognizer using isolated characters of IRONOFF and UNIPEN database separately and in combination of both (the IRONOFF-UNIPEN database). For embedding SVM in the hybrid word recognition system later, we need to retrain it using characters cut from the word database. This is due to the fact that handwritten isolated characters are written differently from characters cut from a cursive word. Isolated characters also have similar origin making it unnecessary to perform comprehensive preprocessing steps required in the characters cut from words.

The block diagram for an online character recognition system is depicted in figure 5.1. The diagram shows the training part on the left and the recognition part on the right. In the training, the character database is used to build the SVM recognizer model. In recognition, single character handwriting is input to the system and the output is given as the text representation of the character. For both training and recognition, the input character signal needs to be preprocessed and normalized before selected feature representation of the character can be extracted. The extracted feature will be fed to the recognizer which has been trained to recognize and produce the letter representing the handwritten input signal. The character recognition system has been trained to recognize characters which are used in English as well as some special characters specific to French. The recognizer was trained using LIBSVM. The result of the training procedure, produce a recognizer represented by an SVM model. SVM model consists of the values for various parameters and the feature values of input characters which have been selected as the support vectors.

We describe the various stages involved in the training and recognition of the SVM recognizer in the following sub sections; namely on signal representation, preprocessing and normalization, feature extraction and training.

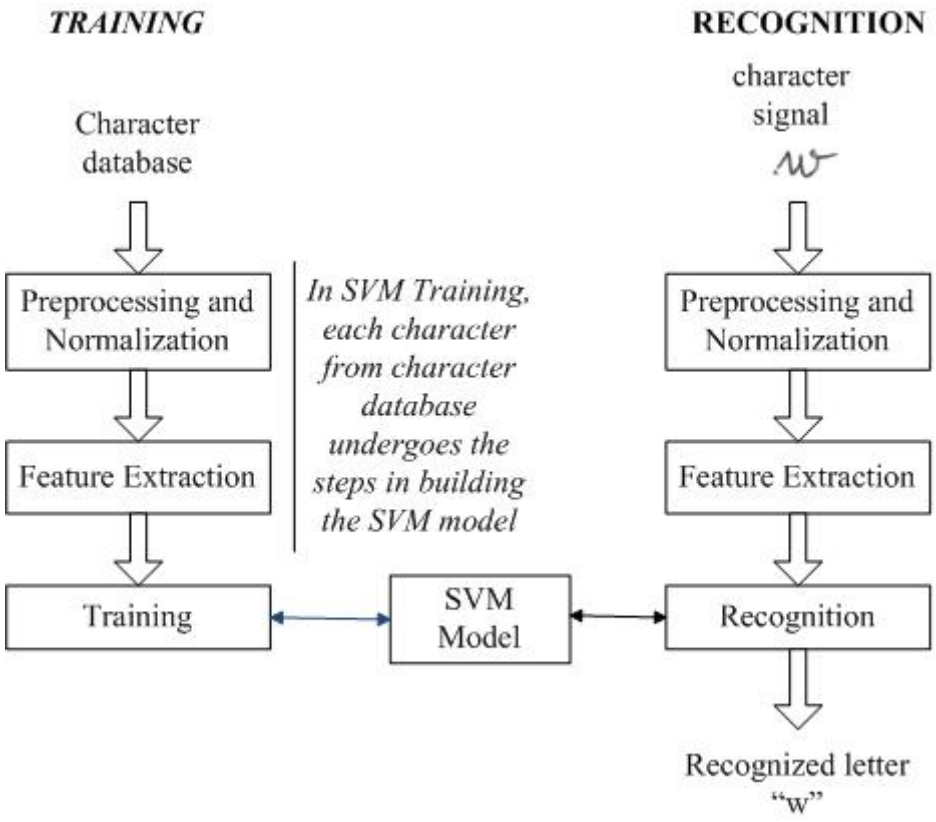


Figure 5.1 Handwritten Character Recognition System

5.2.1 Signal Representation

The input signal to the recognition system is a character signal which consists of a combination of strokes. Stroke is the trajectory traced by the pen from a pen-down event to a pen-up event. For storing the captured online signal into a file, the storage of the signal follows the UNIPEN format description shown in Figure 5.2. UNIPEN format starts with comment lines where description of the writer and the signal to be captured can be recorded. Among the information stored in the comment portion are the signal recording quality in points per inch, dimension of the recording tablet in number of points and the information about the writer. In the data portion, a stroke is a list of coordinate points from after the .PEN_DOWN until a .PEN_UP. Each point in the stroke can record the X and Y coordinates together with the pressure and time information. The number of points in a character signal depends on the number of strokes and the speed of writing.

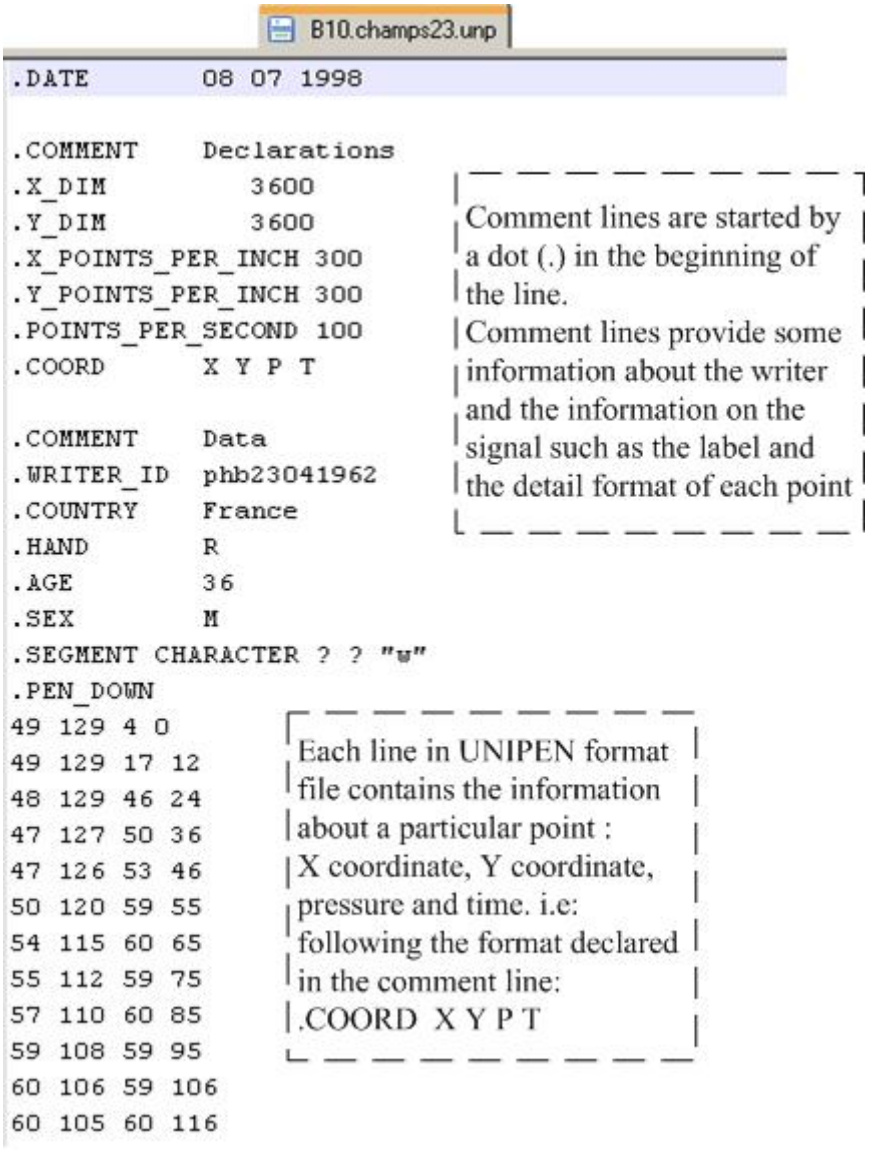


Figure 5.2 Example portion of UNIPEN file showing the format for online handwriting signal

A character can be written with more than one stroke. The total number of strokes and the total number of points in each character varies. This can cause problem to certain type of recognizers. In HMM based recognizer, character representation can be of variable length but SVM requires the feature vector representing the character to be of constant length. We have empirically chosen to use a standard 30 points to represent characters in our experiments. For characters having more or less points, resampling and interpolation was done during the preprocessing stage to standardize the number of points to 30.

5.2.2 Preprocessing and Normalization

Preprocessing of the input signal is done to improve its quality which can lead to better feature representation and recognition. Noise, in the form of repeated points and others due to erratic hand motions and imperfections in the digitization process needs to be eliminated. Then a new signal which has uniform number of 30 equidistant points is obtained by resampling and interpolation of the signal which have been cleaned up from noise. The procedure for resampling is straight forward as in Figure 5.3. In the case of signals with more than a stroke, total length of the signal is taken to be inclusive of the pen-up distances between strokes and imaginary pen-up points are inserted between strokes, i.e.: between the pen-up(s) and pen-down(s). The number of equidistant points is decided based on experimentation done in the effect of recognition accuracy and time of training.

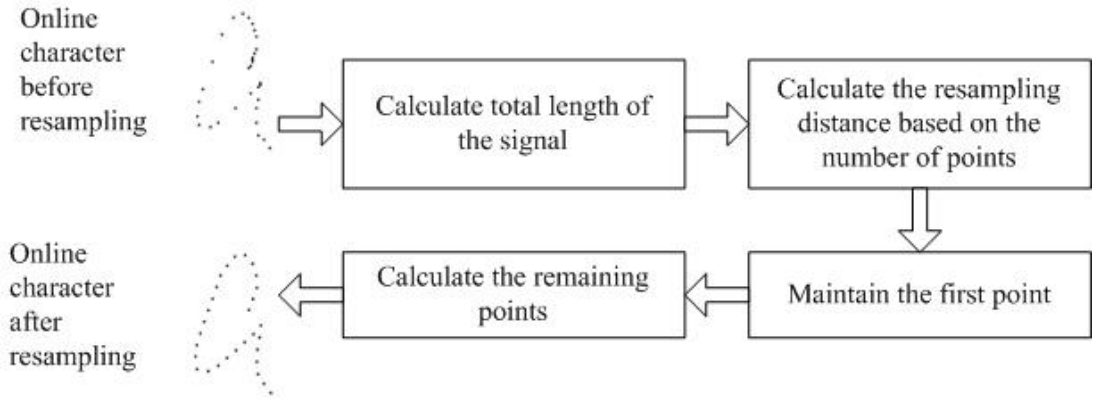


Figure 5.3 Resampling of Online character signal

Signal normalization is a standard procedure in almost every recognition system. In our case, the normalization done on the character signal is mainly to standardize the input signal so as to make it invariant to translation, spatial distortion, character size and style of handwriting. What have been done was to centre the coordinates of the signal and to rescale it according to a rescale factor determined based on the extreme coordinates of the character. Normalization of the character signal is simpler than word normalization which requires the detection of four reference lines; the descender line, the base line, the core line and the ascender line of the word.

5.2.3 Feature Extraction

For training as well as recognition, the resulting signal after preprocessing and normalization is used to extract feature values for use in either the training or recognition. In our case, 7 feature values were extracted for each point resulting in 210 feature values altogether. For all 30 points, the feature values for each point x(n), y(n) are as follows:

(i) Normalized x(n) between -1 and 1.
(ii) Normalized y(n) between -1 and 1.
(iii) Cosine of the direction angle of the line between point x(n+1), y(n+1) and the point x(n-1), y(n-1) and x axis.
(iv) Sine of the direction angle of the line between the point x(n+1), y(n+1) and the point x(n-1), y(n-1) and x axis.
(v) Cosine of the curvature angle between the point x(n+2), y(n+2) and the point x(n-2), y(n-2) at x(n), y(n).
(vi) Sine of the curvature angle between the point x(n+2), y(n+2) and the point x(n-2), y(n-2) at x(n), y(n).
(vii) The binary value of +1 for pen-up or -1 for pen-down.

Features (iii) and (iv) constitute the direction information and features (v) and (vi) constitute the curvature information. Figure 5.4 show in better detail, the 4 features related to directions and curvatures in (iii), (iv), (v) and (vi) above. The 210 features extracted from the 30 points of the input signal are used together with all sample input handwriting to train the SVM. For recognition, the feature values for a single input character are used for its recognition.

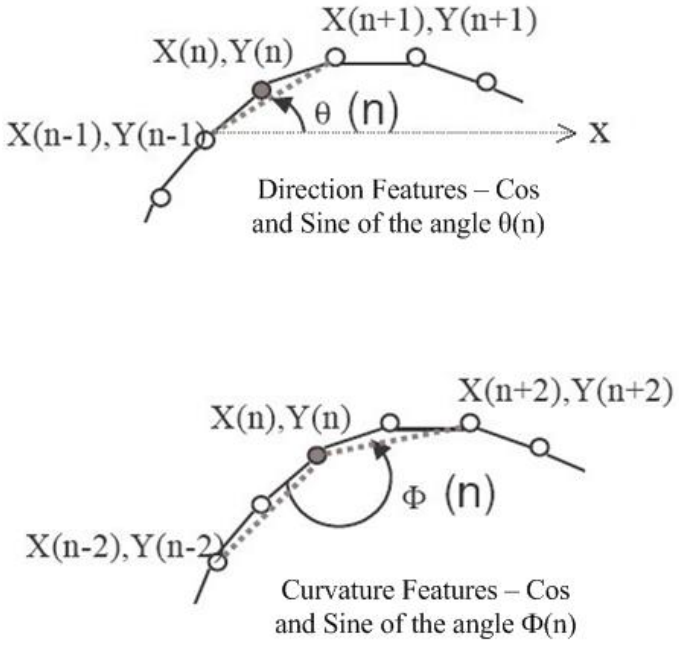


Figure 5.4 Direction features (above) and curvature feature (below)

5.2.4 Training and Recognition

SVM Training was done using all characters in the IRONOFF and UNIPEN databases and the special database which combines the two. Publicly available SVM library packages were compared in order to choose the most suitable for our use. We selected LIBSVM library after conducting some experiments using smaller databases from UCI repository (see section 6.3) and finally on the IRONOFF character databases. The integration of the library into our recognizer is straight forward. However, as most SVM implementer chooses to do, the sparse format representation was used in storing the feature in the process after feature extraction.

The SVM character recognizer classifier equation is given as

$$f(x) = \text{sgn}(\sum_{i=1}^{N} \alpha_i \, y_i K(x_i . x) + b) \qquad \text{(Eq. 5.1)}$$

where the α_i's are the Lagrange multipliers and *b* the bias that are solved during SVM training. The α_i's are either positive or 0. When they are positive, the

corresponding example i contribute in the calculation of the output f of the recognizer. The variables x_i and y_i are the i^{th} example and its label respectively and x is the character to be recognized. K is the kernel which indirectly performs the dot product for the examples in linear high dimension. There are 4 popular kernels used in SVM, as given in Table 4.1, linear, polynomial, Gaussian Radial Basis Function (RBF) and sigmoid .

In our character recognizer training, we have tried both the polynomial kernel $K(x,y)=(\gamma x.y+r)^d$ and Gaussian RBF kernel $K(x,y)=\exp(-\gamma\,|\,x-y\,|^2)$. The reasons why we only choose to try the two kernels are as follows.

(a) RBF kernel and polynomial kernel are nonlinear kernels suitable for the case when the relation between class labels and attributes is nonlinear, as in handwriting recognition.

(b) Linear kernel is just a special case of RBF as the linear kernel gives the same performance as the RBF kernel with some parameters C and γ (Keerthi, 2003).

(c) Sigmoid kernel behaves like RBF for certain parameters (Lin, 2003) and sigmoid kernel is not valid (i.e. not the inner product of two vectors) under some parameters (Vapnik, 1995).

Among the two kernels, Gaussian RBF kernel performs better and we have chosen to use RBF kernel in all our SVM trainings. A few reasons why we finally settle for RBF kernel are as follows;

(a) RBF kernel has less number of hyper parameters than the polynomial kernel, which influences the complexity of the training. However, RBF kernel is not suitable when the number of features is very large. It is more suitable to use the linear kernel. This does not apply to our case as we have reasonably finite feature size.

(b) RBF kernel has less numerical difficulties. One key point is that RBF kernel values are between 0 and 1, while polynomial kernel values may go up to infinity or even 0 when the degree d is large.

For both cases, 10-fold cross-validations were performed in order to search for the best values for the kernel parameter γ and the C values of the SVM. Once the best parameter values were obtained, the character SVM is retrained using those best parameters to obtain the final SVM model to be used in the recognizer. The results of recognition by the SVM character recognizer have been compared with other character recognizers such as TDNN and MLP Neural Networks (Caillault, 2005). The detail results on various databases used are presented in chapter 6.

5.3 The online Word Recognition System

5.3.1 Previous Systems

The general layout of the architecture for the handwritten word recognition system used in our work is depicted in Figure 5.5. It is a segmentation-based recognizer (SegRec) and lexicon-driven which is similar in structure to some other works described in the literature, in particular by (Tay, 2002) and (Caillault, 2005) from the same laboratory.

The work by (Tay, 2002), in off-line handwriting recognition uses hybrid of ANN and HMM for word recognition. The segmentation was INSEG based and the training of the system was using character level discriminant training where the ANN and HMM were independently optimized. Word level discriminant training where the ANN was optimized based on word level output by the HMM were also attempted and compared. In character level discriminant training, junk characters that are formed from the slice combinations which do not resemble any characters need to be handled. In word level training, error in word recognition is back propagated to the ANN using the maximum likelihood (ML) and also the maximum mutual information (MMI) criterions.

(Caillault, 2005) on the other hand worked in online handwriting recognition. She uses OUTSEG segmentation and makes use of a hybrid of Time Delay Neural Network (TDNN) and HMM for word recognition. The TDNN spot characters scanned by equal sized overlapping segment windows over the input handwriting. The character recognizer is used to produce posterior probabilities of characters for each segment and the Viterbi algorithm produces the score of each word from the combination of the segments.

In both cases of (Tay, 2002) and (Caillault, 2005), the recognition score of the input word is compared against all words in the lexicon to obtain an n-best list. The recognition rate was evaluated based on the ranking of correct word recognition at top-n ranks, for example top-3 means the recognition rate in which the word is recognized correctly in either one of the top three positions. In our case, we use a similar system as in (Tay, 2002) where INSEG segmentation is used but instead of offline, we focus on online handwriting recognition. The reason for using INSEG is due to the character level discriminant training approach that we adopted which does not allow OUTSEG segmentation. The main difference which is the centre of our thesis is the use an SVM in place of the ANN for the character recognizer. The training was done at the character level; the reason being that normally, SVM training does not involve correcting gradients like ANN but requires quadratic optimization. (Note: It is observed that Telstra Australia has patented a gradient-based SVM training method for SVM recently (Kowalczyk, 2001)). Table 5.1 compares our system with the other two systems above.

Table 5.1 Comparison of the three handwriting systems developed

Author	Segment. method	Domain	Character Recognizer	Training method	Databases used in testing
(Tay, 2002)	INSEG	Off-line	ANN	Character and word level	IRONOFF, SRTP AWS, MNIST
(Caillault, 2005)	OUTSEG	On-line	TDNN	Word level	IRONOFF, UNIPEN, MNIST
Our work	INSEG	On-line	SVM with RBF kernel	Character level	IRONOFF, UNIPEN, MNIST

5.3.2 General Description of the Hybrid SVM/HMM Word Recognition System.

Figure 5.5 shows the INSEG based hybrid handwritten word recognition system which was developed. It shows a trained system that receives a word to be recognized. Another input to the word recognizer is the list of words or the lexicon containing all the words in the recognition vocabulary. The output of the word recognizer is a list of top N words that resemble the input word signal, in the best resemblance order. A postprocessor can make use of the N-best list for selecting the final word output for the recognizer. The character recognizer used in the overall hybrid word recognizer has been trained optimally using SVM with the best segmented characters from the word database.

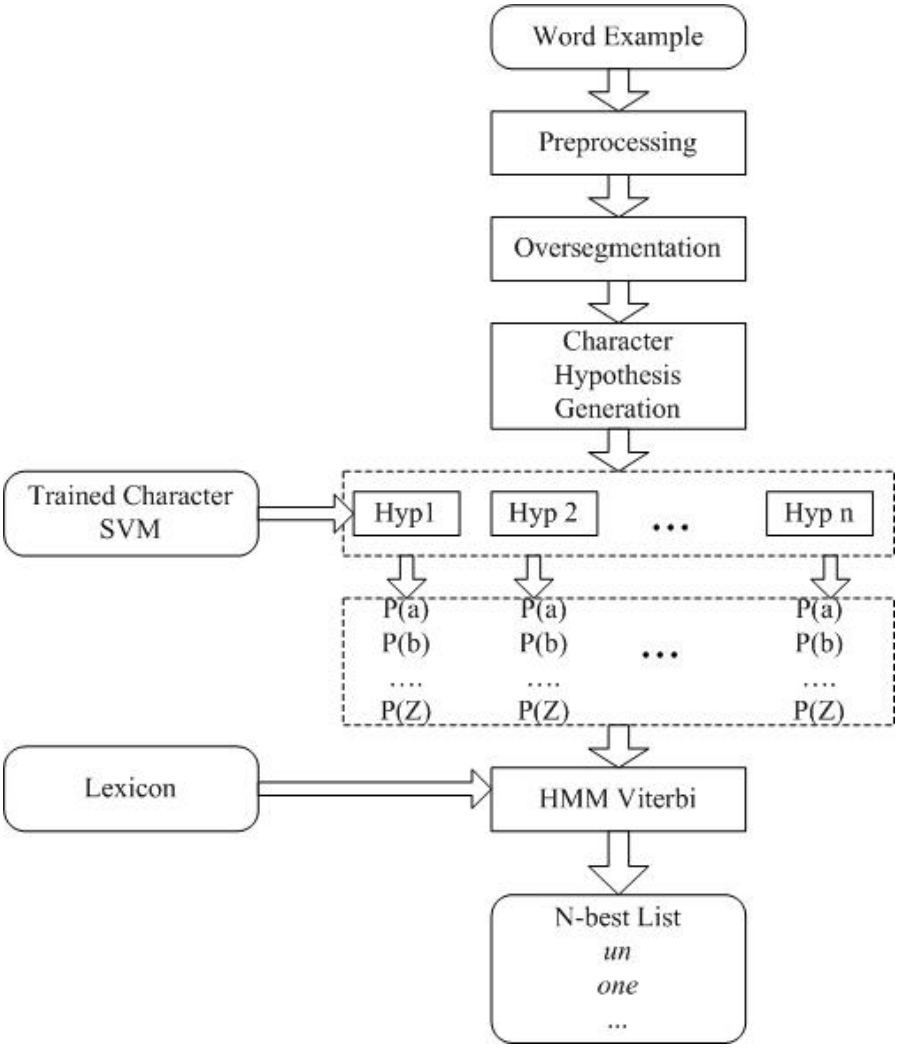


Figure 5.5 The overall hybrid handwriting recognition system

We only make use of IRONOFF online word database for training and testing the hybrid word recognizer. The database is limited to the lexicon that consists of 197 words from the English, French. 30 of the French words are French cheque words. A more detailed discussion of the databases is given in chapter 6.

The initial hybrid SVM/HMM word recognition system needs a trained character recognizer. The character recognizer described in section 5.2 was used but the SVM was retrained using the set of characters which were cut from words in the IRONOFF word databases. This is necessary as isolated characters are different from characters cut from words. First, we used a commercial handwriting recognizer to cut the word database into our initial isolated character database. For testing and comparison purposes, we have trained separately the character SVMs for characters obtained from cheque words, characters obtained from French words and characters obtained from English words. These SVMs were used in separate hybrid word recognizers catering for the respective word types for testing purposes.

In the final system, a single SVM character recognizer was used where the training of it makes use of combined characters from all the separate word databases together. The character SVM recognizer trained with the characters cut by the commercial recognizer is what we call as the bootstrap character recognizer which will be retrained to become a fully trained character recognizer during the training of the hybrid word recognizer.

The training of word recognizer is described in detail in the following sub sections, starting with the front-end portion of the hybrid system. Figure 5.12 portrays the overall training process of our word recognizer which was trained at the character level. The front-end involves preprocessing, segmentation, character hypothesis generation from the segments and feature extraction of the hypothesis. These are followed by the recognition portion of the system which involves the recognition of each character hypothesis by using the SVM character recognizer and the use of Viterbi algorithm or dynamic programming algorithm to find the word score for each word in the lexicon given the input word signal.

The word in the lexicon with the highest score is taken as the recognized word. To evaluate the word recognizer, we also look at the position of the correct word in the top N ranked words. The Viterbi algorithm yields the best segmentation points of the correctly recognized words which can be used to resegment the word database into character database for the next round of Character SVM training.

5.3.3 Preprocessing and Normalization

As can be seen in Figure 5.5, in the overall recognition system, input word signal need to be preprocessed in order to eliminate spurious signals which can have an effect on recognition. The preprocessing step during training stage is done to the whole word database so that input words are already preprocessed words. In a trained recognizer, preprocessing of input word is done within the system, before recognition. Preprocessing involves noise reduction and normalization.

Noise reduction can be done by limiting the bandwidth of the frequency of the data using filtering, where cusps are treated as boundary points to avoid smoothing out important shape features. The cusp detection algorithm captures only dominant cusps while ignoring small wiggles caused by noise. In normalization, geometric variance due to writing style differences among different writers or within the same writer is reduced. Normalization may include scaling of handwriting to a standard size, rotation of the text baseline and deslanting of slanted text. In our system, we performed word rotation correction with reference to the baseline and size normalization as indicated in Figure 5.6. In order to do that, first, the set of maxima and minima points in the word are detected and the four reference lines need to be calculated from the raw input word.

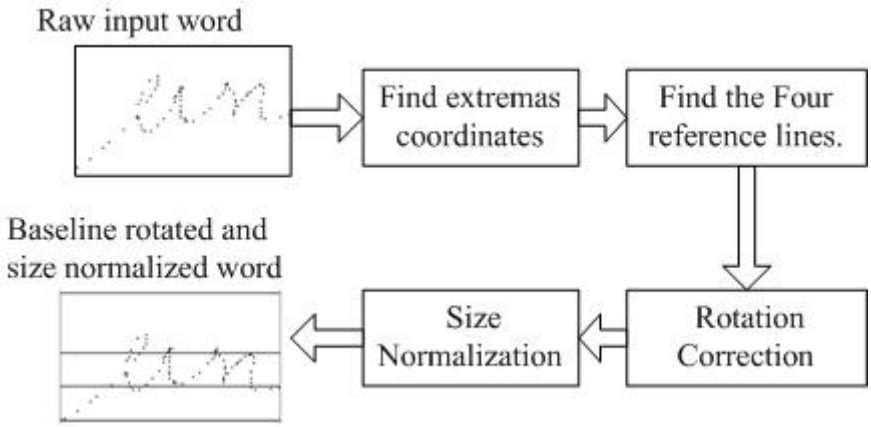


Figure 5.6 Normalization steps in word preprocessing

An example of the set of four reference lines for the word "neuf" is as shown in Figure 5.7. The reference lines are important during both size normalization and rotation correction. They are determined by a simplified algorithm based on Expectation Maximization (EM) algorithm for word normalization in (Bengio, 1994). Together with the a priori probability distribution of the line positions, the maximum and minimum points are used as the observations for the EM algorithm in order to estimate position and rotation angle of the four straight parallel reference lines.

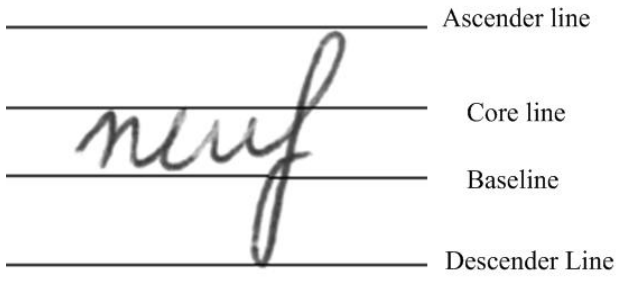


Figure 5.7 The four Reference Lines

In the correction to word rotation angle, the reference lines are brought back to be parallel to the x axis by a rotation of angle α formed by the lines and the x axis. In word normalization, the size of word is normalized to tackle the possibly large variability in the different input that the recognizer is to support. The word is resized to have one unit size for the core line – baseline distance and standardized in both

dimensions so that it will not distort the appearance of the word. The deslanting of slanted text we mentioned earlier as part of word normalization was not done. It is intended that the variation will be adapted and absorbed in the character SVM modeling stage.

One important step of preprocessing as mentioned in chapter 2 is the handling of the diacritical marks, i.e: the crosses of the "t" and "x"s and the dots in the "i" and "j". The diacritical marks can be removed to create a clean word signal without diacritics following (Guyon, 1996). However, we did not perform this step.

5.3.4 Over Segmentation and Hypothesis Generation

In order to generate hypothesis for characters, input word is over segmented into slices. Figure 5.8 shows the over segmentation for the word "*un*". Over segmentation are done based on maximum and minimum y coordinates in the word. This is considered a basic and very simple method. In the figure, the word "un" yields 8 slices as shown above. For longer words, there will be more slices. The slices are combined to form character hypothesis. The total numbers of hypothesis affect the complexity of our training process as well as the accuracy in the recognition. This number, in turn, depends on the minimum and maximum total number of slices to be included in a character hypothesis. The parameters; minimum and maximum number of slices were decided heuristically based on trial and error but supported by the knowledge of the average number of maxima and minima points that generally exist in a character. In order to create proper character hypothesis, the number of slices to combine is very important.

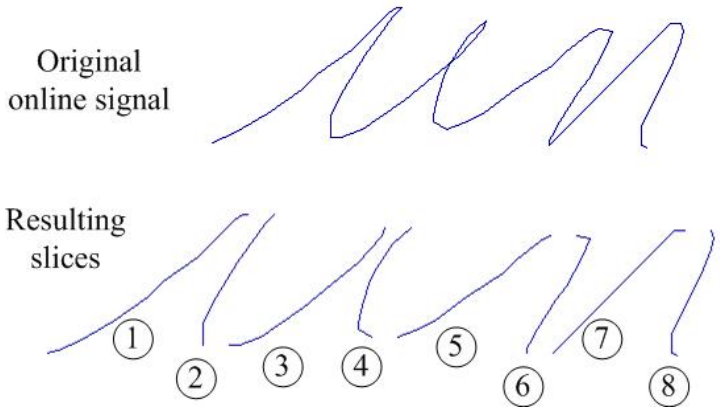


Figure 5.8 Oversegmentation of the word "un" based on minimum and maximum y points

The total number of hypothesis can be calculated by the following formula:

```
if (num < max)
   tot=(num-min+1)*(num-min+2)/2;
else
   tot=(num-max)*(max-min+1)+(max-min+1)*(max-min+2)/2;
```

where *tot* is the total number of hypothesis generated, *num* is the number of slices, *min* and *max* are the minimum and maximum slices in a hypothesis respectively. For the sake of illustration and clarity purposes, an example of slicing and hypothesis generation for offline handwriting is given in Figure 5.9 (It is easier to draw the diagram for offline compared to online). Here, *num* is 5, and assuming *min* is 1 and *max* is 3, *tot* is (5 - 3) * (3 – 1 + 1) + (3 – 1 + 1) * (3 – 1 + 2) / 2 = 2 * 3 + 3 * 2 = 12. Similarly, the total number of hypothesis for the example in Figure 5.8 is then (8 - 3) * (3 – 1 + 1) + (3 – 1 + 1) * (3 – 1 + 2) / 2 = 5 * 3 + 3 * 2 = 21.

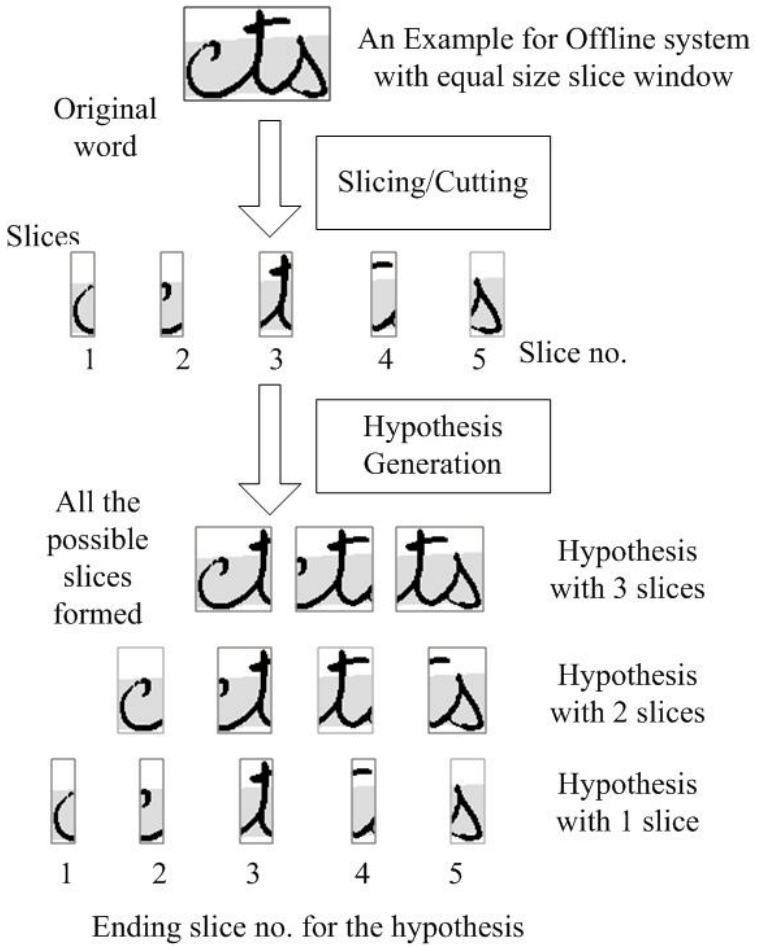


Figure 5.9 Character Hypothesis Generation: A simple example for offline in slicing and generating hypothesis using the word "cts", assuming 5 slices.

The maximum number of slices for a character is chosen based on statistics on the training dataset. We found that a maximum value of 7 and minimum of 1 to be suitable since a lower case letter on average contains 5 slices. So as to cover all characters, we have taken into account the jaggedness at the beginning and within the writing, and choose the maximum value of 7. This has also been verified experimentally.

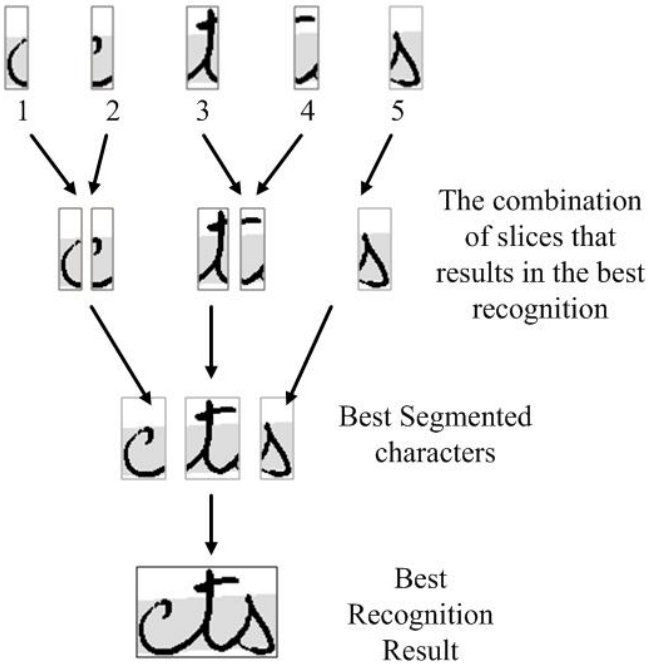


Figure 5.10 Result of recognition and Segmentation

In order to form a word, only the related non-overlapping hypotheses are used. The job of choosing the correct hypotheses which means correct character segmentation lies in the dynamic programming algorithm in the HMM. Figure 5.10 shows an example of the best segmentation which resulted in the best recognition.

5.3.5 Feature Extraction

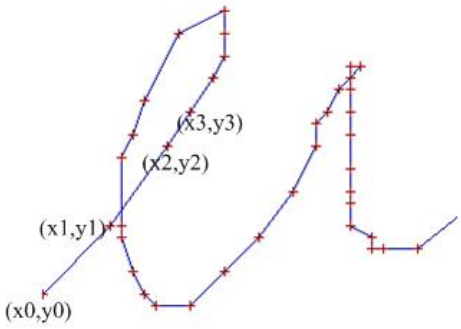
Since our system caters for training of the SVM character recognizer, the features extracted are for each of the hypothesis character that have been formed by joining the over segmented slices. Before feature extraction, signal resampling is done on the hypothesis character to standardize the number of points in the character signal to 30 points. The number 30 is selected based on heuristics with the aim of having smaller feature dimensions. Note that our word recognizer did not perform resampling at the word signal since we are doing the resampling at the character stage similar to what has been done in isolated character recognizer.

We extract seven features per point as in the isolated character recognizer. However, since each character is now cut from a word, the first of the seven features, the x coordinates will be different from x coordinates of isolated characters. The original x coordinates undergo a translation relative to the beginning of the character. This in effect adjusts the x character coordinates to start from a common point of zero. Figure 5.11 shows the first 4 coordinates of the original character u and the new coordinates calculated for the feature values. The new x coordinates for each point in the character hypothesis will be as follows:

$$x_{offset} = x_0 \qquad \text{(Eq. 5.2)}$$

$$x_n = x_n - x_{offset} \qquad \forall n$$

The other 6 features are essentially the same as in the isolated character recognizer, which are; the y coordinates, the 2 direction features, 2 curvature features and the pen-up/pen-down information (see section 5.2.3). The output of feature extraction stage is a sequence of vectors containing the 210 required features, i.e: 7 features for the total of 30 points. The vector sequences are then provided as input to the SVM character recognizer.



n	x_n	y_n	New x_n	New y_n
0	164	110	0	110
1	170	104	6	104
2	175	97	11	97
3	177	94	13	94

Figure 5.11 Example of new x values for the hypothesis character. Shown in the table - only the first 4 points. Y coordinates remain.

5.3.6 Overview of Hybrid SVM/HMM Training

The training process for the word recognizer is given in Figure 5.12. The whole aim of the training is to optimize the character SVM by using the set of characters

which are the best segmented from the word as a result of the recognition. The quality of the word recognizer relies on the quality of the SVM character recognizer, which in turn relies on the segmentation made on the words into characters. Initial training uses the bootstrap SVM character recognizer. The objective in training of the system is then to try to improve further the initial bootstrap SVM character recognizer by improving the segmentation of the input word while recognizing it.

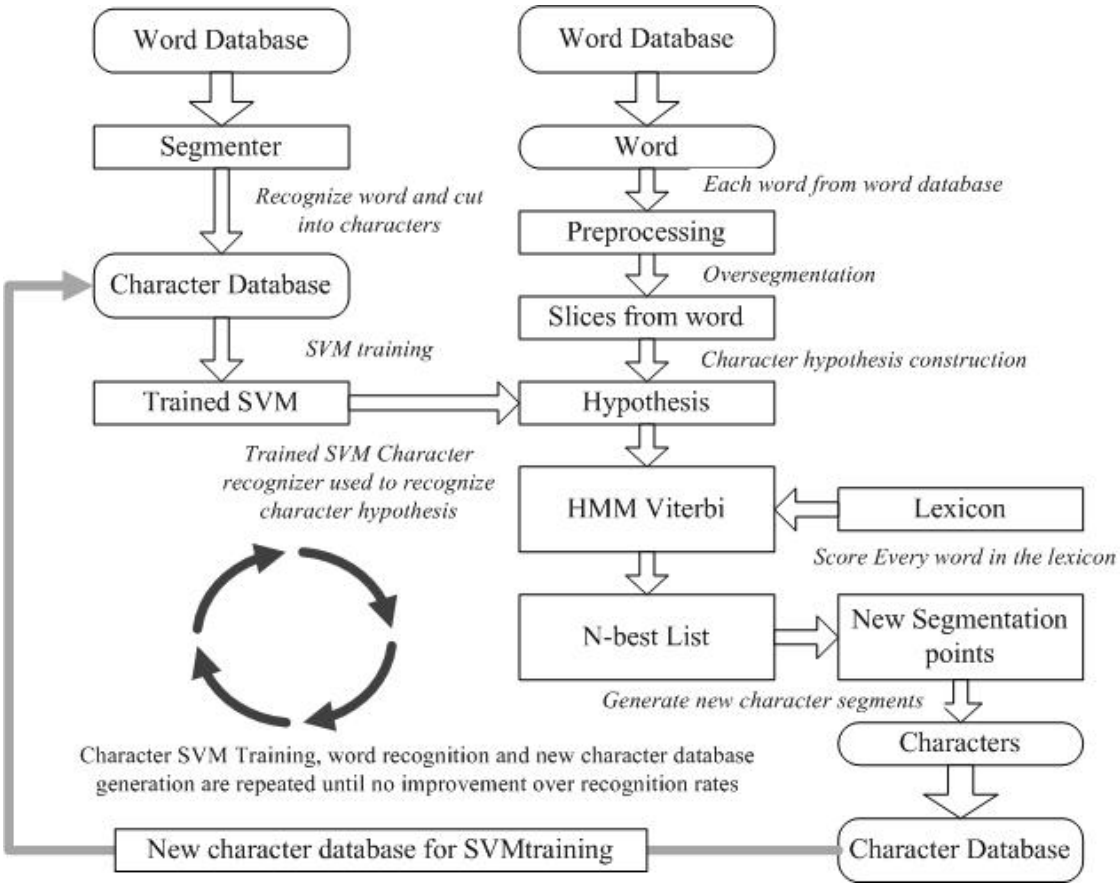


Figure 5.12 Character level training for word recogntion system

A hypothesis character is recognized by the trained SVM character recognizer and assigned a set of confidence values. There is one confidence value for each character class. The confidence values reflect the degree to which the primitive or union of primitives represents that class. For each word in the lexicon, dynamic programming or the Viterbi algorithm is used to find the best sequence of hypothesis character to match the word input signal using the confidence values. At the end of recognition of each input word, new segmentation points (if any) are determined by

the word recognizer output following the segmentation that gives the best word score.

New segmentation points obtained determine a new and better set of segmented characters to be used in the character database generated for SVM retraining. Better segmentation generate better quality character database and eventually improve the quality of the SVM character recognizer in subsequent training. As character recognizer improves, it is hopeful that the new word recognizer will improve further and the resulting segmentation points will also be better. This cycle of improving character segmentation and retraining of SVM is repeated a few times until there are no more improvements in the word recognition rate.

For training the word recognizer, we used word signals that have been preprocessed in a separate preprocessing step resulting in a new database of preprocessed words. However, for recognition, since raw word signals are given to be recognized, preprocessing is done within the real time recognition module just prior to recognition process.

5.3.7 Word Likelihood Computation

One crucial issue in training or recognition is the word likelihood computation. Given a word signal to recognize or the word observation O, and a lexicon of words, the word that is taken as the recognized word, $\hat{W}$ is the one that has the highest score among all the words.

$$\hat{W} = \arg\max_{W} P(W \mid O) \qquad \text{(Eq. 5.3)}$$

To calculate word likelihood for each word, we can formulate the problem in the hybrid SVM/HMM framework. The HMM is a left-right model with unity transition probability throughout. Word HMM is a concatenation of several character HMMs.

The hybrid framework uses SVM to compute the observation probability at each state in the word HMM. We can use either the Forward-backward or Viterbi algorithm to calculate the word likelihood. Word likelihood computation using the Forward-backward algorithm involves summing the likelihoods through all the possible paths for the particular word of the lexicon. However, Viterbi algorithm which finds the single best path is a good approximation that we use. Forward-backward and Viterbi algorithm falls under dynamic programming approaches.

The problem can also be straight forwardly approached as a search problem, without involving HMM framework. This is possible as we use a simple left-right HMM with unity state transition, which means that the emission probability is the only visible component in the calculation of the likelihood along the path; the emission probability being the character likelihood generated by the SVM. In this manner, other time saving search methods could be used such as beam search (Ney, 1987), heuristic search etc.

Figure 5.13 shows in a general framework, the processes involved in word likelihood computation. Each character hypothesis is passed through the feature extractor and SVM character recognizer, which will output the probability scores for all the labels in the SVM model, in which it has been trained. Since the characters are cut from the words that we are training the word recognizer with, all character labels from the database should be covered by the SVM. Given all the arrays of character probabilities for each hypothesis, to calculate the word likelihood for a particular word in the lexicon, the Viterbi or the forward-backward algorithm is used to sum all the log values of character likelihood across allowable path which does not contain hypothesis which overlapped each other. The path shown at the bottom of the diagram through 3 hypotheses, the first which contains slice 1 and 2 followed by the second which contains slices 3 and 4 and finally the one that contains only slice 5 forms the best path.

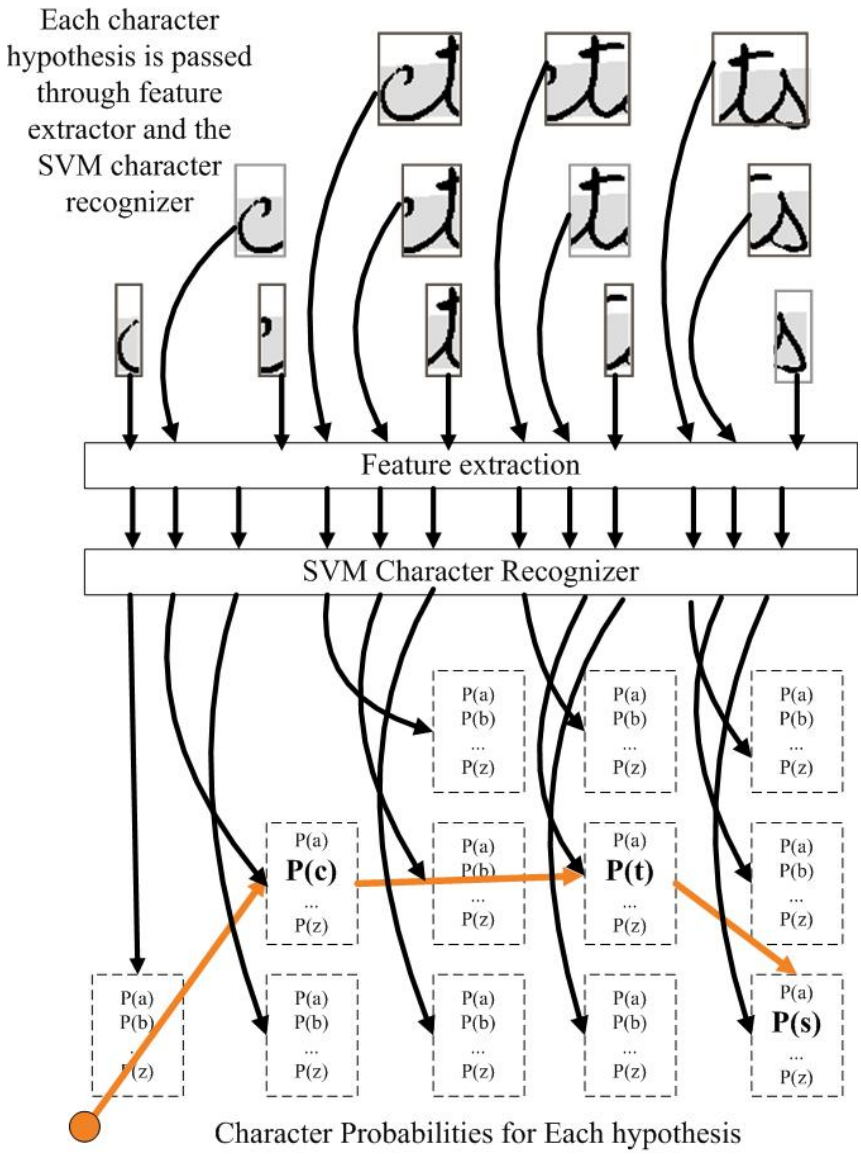


Figure 5.13 Word Likelihood Computation – The best word is "cts", through slice combination 1 & 2 for char c, 3 &4 for char t and slice 5 for character s. Bold and large P(*i*) indicates largest probability values for character *i* .

5.3.8 SVM/HMM Framework

As we mentioned earlier, we can understand the recognition system, in particular the word likelihood computation by putting the above in an SVM/HMM hybrid framework. At the character level, each character can be represented by a character HMM. We have considered 68 character HMMs to represent 68 letters and symbols. Out of that, 26 are for small letters, 26 for capital letters and 4 for symbols, for a total of 56. Since our databases cater for English and French words, 12 extra HMMs

for letters special to French are used, making the total of 68. All the 68 letters and symbols are listed in Table 5.2.

Table 5.2 The 68 Character HMMs

a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	Y	z	A	B	C	D
E	F	G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V	W	X
Y	Z	-	'	.	,	à	â	ç	é
è	ê	ë	Î	Ï	ô	ù	û		

An example of a character HMM is given in Figure 5.14. Note that there is an entry node and an exit node which are used to concatenate between character HMMs.

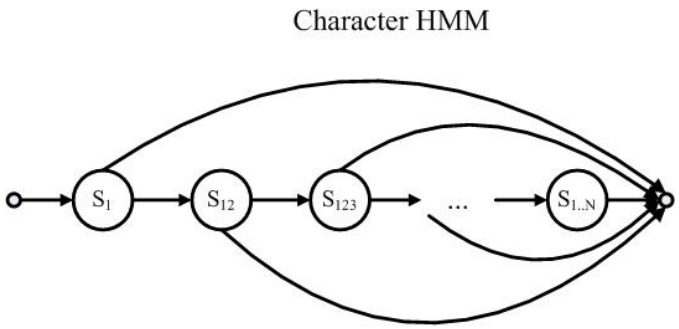


Figure 5.14 An example character HMM with N states

The character HMM topology is a left-right topology with N states where N is the maximum number of slices allowable for each character. The n^{th} hidden state represents the state where the character emitted is made up of *n* slices from starting slice up to the n^{th} slice. For example at state S_{12} the character emitted consists of 2 slices; 1 and 2. Once a character is emitted, transition will proceed to the exit state. If a character is not emitted, transition will be to the next state. The transition probability for the character HMM is set to 1 since we allow the HMM to move between states from left to right and to exit state with equal probability. The

emission probability is given by the SVM character likelihood score. Character HMMs are concatenated to form word HMM as shown in Figure 5.15.

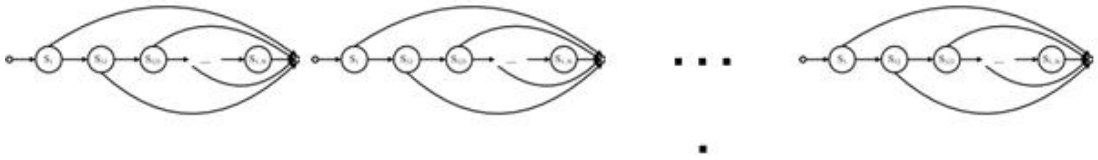


Figure 5.15 Word HMM formed by concatenating character HMM

Figure 5.16 shows an example of the word recognition graph for recognizing the word "cts". For *T* number of slices, the observation sequence is $O= O_1O_2O_3,...,O_5$. In our SVM/HMM framework, we need to cater for each observation to have a maximum of N slices representing each hypothesis that is made up of slices ending with slice t. (In the diagram, T= 5, N = 3). The likelihood of each word-HMM, λ given observation sequence O, or $P(O \mid \lambda)$, can be computed as below:

$$P(O \mid \lambda) = \sum_{\Gamma} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(O_t) \qquad \textbf{(Eq 5.1)}$$

where we use the transition probabilities $a_{q_{t-1}q_t}$ of 1.

We use the Viterbi algorithm which gives a single best state sequence as discussed in section 3.2.5, to estimate the word score. We define the quantity $\delta_t(i)$ which represents the best score along a single path, at time *t*, which accounts for the first *t* observations and ends in state *i* as follows:

$$\delta_t(i) = \max_{q_1,q_2,\cdots q_{t-1}} P(q_1q_2q_3\cdots q_t = i, O_1O_2O_3\cdots O_t \mid \lambda) \qquad \textbf{(Eq 5.2)}$$

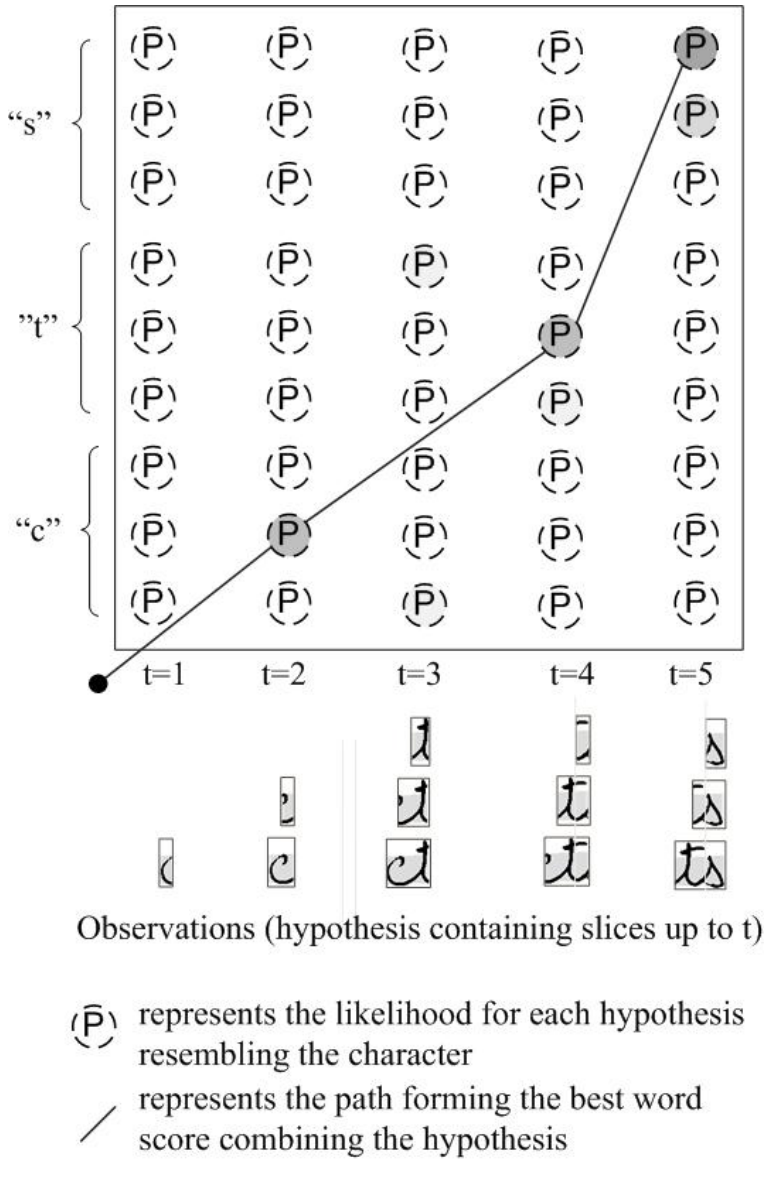


Figure 5.16 Word Recognition Graph

By induction, we have

$$\delta_t(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}) \qquad \textbf{(Eq 5.3)}$$

The complete procedure to compute the word score using the Viterbi algorithm is :

a) Initialization

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \qquad \textbf{(Eq 5.4)}$$

b) Recursion

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}] \cdot b_j(O_t), \quad 1 \le j \le N \qquad \textbf{(Eq 5.5)}$$

c) 3. Termination

$$P^* = \max_{1 \le i \le N}[\delta_T(i)] \qquad \textbf{(Eq 5.6)}$$

In the actual word score calculation, we made use of the log values of the observation probability, which turns the multiplication in the formula into addition. Figure 5.16 shows that the best path for the Viterbi algorithm as using character 'c' from hypothesis containing slices 1 and 2, then for character 't' from hypothesis containing slices 3 and 4 and finally 's' from hypothesis containing only slice 5. The circles with dark shades in the diagram are the hypothesis with the optimum log probability values which added together in the best path giving the best score.

5.4 Summary

In this chapter, the handwriting recognition being developed is discussed. We presented first, the SVM handwritten character recognizer, from the preprocessing stage through to normalization, feature extraction and training. The hybrid SVM/HMM word recognizer is then described from the same level of perspective. Finally, detail discussion on word likelihood computation is presented, making use of recognition graph and the Viterbi algorithm.

CHAPTER 6

DATABASE AND EXPERIMENTAL RESULTS

6.1 Introduction

We have developed a complete online handwriting recognition system that implements the hybrid of HMM and SVM which is the focus of this thesis. There are various issues that need to be addressed in order to make the implementation of the system successful. To test the system, we have conducted a number of experiments. We made use of some available databases for the experiments to test the validity and usefulness of our system at various stages of implementation.

This chapter describes the databases that were used and the various experiments that have been performed. We first describe general public databases which were used to evaluate our SVM. There are a few subsets of the UCI datasets (Newman, 1998) that we used in our very early experiments on SVM. Then we focus on the two main handwriting databases that have been used; the IRONOFF database (Viard-Gaudin, 1999) and the UNIPEN database (Guyon, 1994). IRONOFF consists of both online and offline data while UNIPEN provide only the online data. The online data of IRONOFF database was collected using the UNIPEN format. A combination of both UNIPEN and IRONOFF online databases was also generated and used in some experiments. We have also used the MNIST database (LeCun, 1998a) in our SVM experiments. We describe the measure of performance that we used in this thesis and the experiments conducted together with their results.

6.2 Databases

In the early stage of adapting SVM into the system, we have to decide on the implementation route of SVM. As we have mentioned, there are numerous SVM packages which can be used and adapted. We begin by performing tests on these SVM packages on simple datasets from UCI repository in order to choose the best implementation. Once an implementation have been decided and used within the hybrid system, we then use the handwriting databases as listed in Table 6.2 for experiments involving handwriting recognition at character or word level. It is important to perform testing on public and widely available databases because the results obtained are more authentic and comparable.

6.2.1 Data From UCI Repository

UCI repository site at University of California, Irving provides various databases for evaluating learning algorithms. Currently, the repository contains over 173 different data sets as described in (Newman, 1998) and (Asuncion, 2007). Among the popular data sets, only Wisconsin Breast Cancer (WBC), Cleveland Heart Disease (CHD), Tic Tac Toe (TTT), Votes (VT) and Handwritten Digits (HWD) were used to compare and select the SVM implementation packages. Table 6.1 gives a summary of the datasets.

Table 6.1 Sample UCI Data Sets

Datasets	Features
Wisconsin Breast Cancer (WBC)	2 class, 569 samples, 30 features
Cleveland Heart (CHD)	5 class, 297 samples, 13 features
Tic Tac Toe (TTT)	2 class, 958 samples, 9 features
Votes (VT)	2 class, 435 samples, 16 features
Handwritten Digits (HWD)	10 class, 3826 samples 64 features

6.2.2 IRONOFF Online and Offline Databases

The IRONOFF database is collected by Viard-Gaudin (Viard-Gaudin, 1999) from IRESTE (currently known as IRCCyN at Ecole Polytechnique de l'Université de Nantes). It contains both online and offline handwriting data. The database contains in both formats, the following; 4,086 isolated digits, 10,685 isolated lower case letters, 10,679 isolated upper case letters together with 410 EURO signs, 31,346 isolated words from a 197 word lexicon. The isolated words comprises both French and English words (28, 657 French words and 2,689 English words). Table 6.2 gives a summary for all handwriting databases used, including the IRONOFF databases.

Table 6.2 Handwriting Databases

(a) Character databases

Type	Database Name	Detail Type	Training Examples	Test Examples	Total
Character	IRONOFF	Digit	3059	1510	4086
		Lowercase	7952	3916	10685
		Uppercase	7953	3926	10679
	UNIPEN-IRONOFF	Digit	13451	6270	19721
		Uppercase	42778	20172	62950
		Lowercase	25662	11621	37283
	UNIPEN	Digit	10423	5212	15635
		Uppercase	34844	17423	52267
		Lowercase	17736	8869	26605
	MNIST	Digit	60000	10000	70000

(a) Word databases

Type	Database Name	Detail Type	Training Examples	Test Examples	Total
Word	IRONOFF	Cheque word	7956	3978	11934
		English Word	1793	896	2689
		French Word	19105	9552	28657

The database was collected from about 700 different writers, mainly of French nationality. Although the database contains both on-line and off-line information of the handwriting signals, only the on-line information is used for our experiments. The on-line data has been sampled with a spatial resolution of 300 dpi and a sampling rate of 100 points per second on an A4 sized tablet. The database is

available as ASCII files written in the UNIPEN format. Personal information of the writer for each sample such as sex, age, nationality, and whether the person is left or right handed are also provided.

The database is divided exclusively into training set and test set. The scriptors of the two data sets are different, reflecting an omni-scriptor situation in which some types of handwriting styles are only available during the training of the system and not available in testing. Table 6.3 shows the complete lexicon of 197 words from IRONOFF database.

Table 6.3 List of words in the IRONOFF lexicon

Un	*rugby*	*secouraient*	*Vacances*	*elle*	*ton*
deux	*jusque*	*monétaires*	*Week-end*	*nous*	*son*
trois	*chômé*	*malversation*	*Xénophobie*	*vous*	*si*
quatre	*vodka*	*pédalerions*	*Yaourt*	*mais*	*une*
cinq	*gîtes*	*compagnies*	*Zénith*	*où*	*même*
six	*whisky*	*pivoteras*	*Apple*	*donc*	*notre*
sept	*oeuvre*	*surgelées*	*Between*	*or*	*votre*
huit	*voilà*	*fréquemment*	*Capability*	*ni*	*leur*
neuf	*zèbre*	*fredonner*	*Directory*	*car*	*entre*
dix	*dépôt*	*moissonner*	*Earth*	*puis*	*on*
onze	*quelqu'un*	*polygonale*	*Fuzzy*	*ne*	*sur*
douze	*vêtir*	*père-noël*	*Giving*	*pas*	*sous*
treize	*gâchez*	*frapperions*	*Hydrogen*	*à*	*plus*
quatorze	*figeront*	*Agglomération*	*Island*	*au*	*moins*
quinze	*buvez*	*Boîtier*	*Job*	*de*	*avec*
seize	*taxis*	*Citoyen*	*Ku-Klux-Klan*	*du*	*ainsi*
vingt	*fjord*	*Démocratie*	*Liberty*	*des*	*qui*
trente	*dégâts*	*Encouragement*	*Money*	*dans*	*que*
quarante	*jazz*	*Fréquence*	*North*	*en*	*quoi*
cinquante	*buggy*	*Gymnase*	*Obvious*	*par*	*quel*
soixante	*impôts*	*Hôpital*	*Parking*	*chez*	*quelle*
cent	*conçu*	*Imperméable*	*Quiz*	*pour*	*quand*
mille	*aïeux*	*Journal*	*Rabbit*	*le*	*tout*
million	*fonça*	*Kiosque*	*Smooth*	*la*	*tous*
francs	*galette*	*Littérature*	*T-shirt*	*les*	*aussi*
centimes	*flûte*	*Maître*	*User*	*ce*	*dont*
euros	*accident*	*Neptune*	*Voice*	*cet*	*dès*
et	*abbaye*	*Occident*	*Warehouse*	*cette*	*autre*
frs	*éclabousser*	*Psychologue*	*X-ray*	*ces*	
cts	*déposerait*	*Quittance*	*Yuppie*	*cela*	
repêché	*thermonuclé*	*République*	*Zero*	*ceci*	
blâmez	*aire*	*Société*	*je*	*celle*	
affût	*sculpterai*	*Température*	*tu*	*celui*	
l'élève	*organisme*	*Urgence*	*il*	*mon*	

The experiments performed on the IRONOFF database can be divided according their categories namely; Check words, French words and English words. Table 6.4 Table 6.5 and Table 6.6 give the lexicons for the three categories respectively.

Table 6.4 Words in the Check Word lexicon (30 words)

Un	*six*	*onze*	*Seize*	*soixante*	*centimes*
deux	*sept*	*douze*	*vingt*	*cent*	*euros*
trois	*huit*	*treize*	*trente*	*mille*	*et*
quatre	*neuf*	*quatorze*	*quarante*	*million*	*frs*
cinq	*dix*	*quinze*	*cinquante*	*francs*	*cts*

Table 6.5 Words in the French Word lexicon (171 words)

un	*cts*	*galette*	*Hôpital*	*ni*	*si*
deux	*repêché*	*flûte*	*Imperméable*	*car*	*une*
trois	*blâmez*	*accident*	*Journal*	*puis*	*même*
quatre	*affût*	*abbaye*	*Kiosque*	*ne*	*notre*
cinq	*l'élève*	*éclabousser*	*Littérature*	*pas*	*votre*
six	*rugby*	*déposerait*	*Maître*	*à*	*leur*
sept	*jusque*	*thermonucléaire*	*Neptune*	*au*	*entre*
huit	*chômé*	*sculpterai*	*Occident*	*de*	*on*
neuf	*vodka*	*organisme*	*Psychologue*	*du*	*sur*
dix	*gîtes*	*secouraient*	*Quittance*	*des*	*sous*
onze	*whisky*	*monétaires*	*République*	*dans*	*plus*
douze	*oeuvre*	*malversation*	*Société*	*en*	*moins*
treize	*voilà*	*pédalerions*	*Température*	*par*	*avec*
quatorze	*zèbre*	*compagnies*	*Urgence*	*chez*	*ainsi*
quinze	*dépôt*	*pivoteras*	*Vacances*	*pour*	*qui*
seize	*quelqu'un*	*surgelées*	*Week-end*	*le*	*que*
vingt	*vêtir*	*fréquemment*	*Xénophobie*	*la*	*quoi*
trente	*gâchez*	*fredonner*	*Yaourt*	*les*	*quel*
quarante	*figeront*	*moissonner*	*Zénith*	*ce*	*quelle*
cinquante	*buvez*	*polygonale*	*je*	*cet*	*quand*
soixante	*taxis*	*père-noël*	*tu*	*cette*	*tout*
cent	*fjord*	*frapperions*	*il*	*ces*	*tous*
mille	*dégâts*	*Agglomération*	*elle*	*cela*	*aussi*
million	*jazz*	*Boîtier*	*nous*	*ceci*	*dont*
francs	*buggy*	*Citoyen*	*vous*	*celle*	*dès*
centimes	*impôts*	*Démocratie*	*mais*	*celui*	*autre*
euros	*conçu*	*Encouragement*	*où*	*mon*	
et	*aïeux*	*Fréquence*	*donc*	*ton*	
frs	*fonça*	*Gymnase*	*or*	*son*	

Table 6.6 Words in the English Word lexicon (26 words)

Apple	*Fuzzy*	*Ku-Klux-*	*Parking*	*User*	*X-ray*
Between	*Giving*	*Klan*	*Quiz*	*Voice*	*Yuppie*
Capability	*Hydrogen*	*Liberty*	*Rabbit*	*Warehouse*	*Zero*
Directory	*Island*	*Money*	*Smooth*		
Earth	*Job*	*North*	*T-shirt*		
		Obvious			

The handwritings in the word database are unconstrained. It contains the variations mentioned in Figure 1.8 and Figure 1.9. This can be seen in the randomly selected sample words "centimes" and "capability" shown in Figure 6.1.

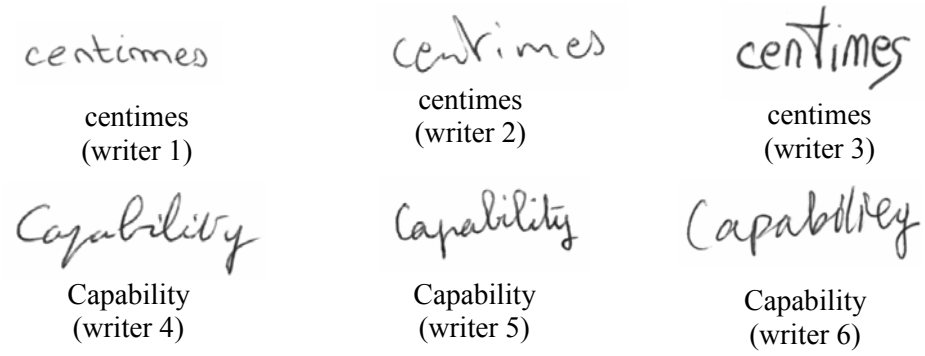


Figure 6.1 Random examples from the IRONOFF Database

6.2.3 UNIPEN Online Character Database

UNIPEN online database (Guyon, 1994, Ratzlaff, 2003) is a database made available by the International Unipen Foundation. The publicly available data sets are named the UNIPEN Train-R01/V07 distribution while there is another set which is not publicly available called the DevTest-R01/V02 subset. Train-R01/V07 are available in 6 categories; namely 1a, 1b, 1c, 1d, 2 and 3. Table 6.7 gives an overview of the overall UNIPEN benchmark database. Table 6.8 gives the detail of the dataset in the UNIPEN Train-R01/V07 distribution.

Table 6.7 UNIPEN Benchmark Overview

Benchmark	Description
1a	Isolated digits
1b	Isolated upper case
1c	Isolated lower case
1d	Isolated symbols (punctuations etc.)
2	Isolated characters, mixed case
3	Isolated characters in the context of words or texts
4	Isolated printed words, not mixed with digits and symbols
5	Isolated printed words, full character set
6	Isolated cursive or mixed-style words (without digits and symbols)
7	Isolated words, any style, full character set
8	text: (minimally two words of) free text, full character set

Table 6.8 UNIPEN Train-R01/V07 Dataset

Category	Type	Number of classes	Total samples
1a	Digits	10	15635
1b	Uppercase letters	26	28069
1c	Lowercase letters	26	61360
1d	Punctuations and other symbols	32	17286
2	Mixed	94	122668
3	Mixed	94	67352

Several authors have published recognition results using this or related UNIPEN databases, thereby affording researchers some reference points for comparing the performance of their recognizers. We have further made a sub selection from the Train-R01/V07 distribution database for our use in the experiments. In particular we only made use of Dataset 1a, 1b and 1c for the digits, lowercase characters and uppercase characters. The actual subset of that we used is given as part of Table 6.2.

6.2.4 IRONOFF-UNIPEN Databases

A combination of IRONOFF and UNIPEN database called IRONOFF-UNIPEN is created and used in our experiments. The IRONOFF-UNIPEN combination basically consists of characters from IRONOFF character data sets and UNIPEN

character data sets. Selection of data from each database is made randomly. This data set has been used to compare character recognition by SVM with other methods, in particular using back propagation neural network (BPNN) and convolutional neural network (CNN) whose results we compare in this thesis. The detail of the database that we used is given as part of Table 6.2

6.2.5 MNIST

MNIST is a modified version of handwritten digit database from National Institute of Standard and Technology (NIST) compiled by LeCun (LeCun, 1995) (LeCun, 1998a). MNIST database is an off-line database. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger data set available from NIST. The digits have been size-normalized and centered in a fixed-size image. The original black and white (bi-level) images from NIST were size-normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain gray levels because of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image to position this point at the center of the 28x28 field. MINIST database have been popularly used by many authors to compare between various machine learning algorithms. We have used MNIST database to compare our SVM implementation with other methods in the literature.

6.3 Experiments in Selecting an SVM package

As mentioned in chapter 4, there are many publicly available SVM packages made available by researchers and implementers. We have tested those implementation using libraries implemented in C or C++ code. In this section we report the results of the tests on three most widely used SVM packages; SVMTorch, SVMlight and LIBSVM. A few experiments were conducted using the three packages. The aim of the experiments is to compare them, using common training parameters and common datasets so that conclusions can be made on suitability, speed of training, accuracy and most importantly ease of use. The packages that we

used during the initial evaluation of SVM, were two class SVM for SVMLight and SVMTorch and multi-class SVM for LIBSVM.

All the raw datasets were processed to obtain input files in the format needed for each package. In each case, the datasets were trained using the different kernels with different hyper parameters and penalty value C ranging from 0.01 to 100. The speed of training, numbers of support vectors, training accuracy and testing accuracy were observed. In the comparisons, we only use RBF and polynomial kernels, for the reasons we give in section 5.2.4. Table 6.9 to Table 6.12 reports the results of the various comparisons that have been made.

6.3.1 Comparing Training Time and Number of Support Vectors

Table 6.9 Training Results for WBC data set (2 class)

SVM Tools	Kernel	C	Number of iteration	nSV
LIBSVM	RBF	1	1128	569
		10	1265	569
	Polynomial	1	490554122	32
		10	623912950	32
SVMLight	RBF	1	121	569
		10	121	569
	Polynomial	1	1237380	34
		10	1062198	34
SVMTorch	RBF	1	840	561
		10	952	561
	Polynomial	1	N/A	32
		10	N/A	32

Note: nSV – number of support vectors
N/A – very large number of iterations.

Table 6.9 gives the training result comparison for all three tools on WBC dataset with C = 1 and C = 10. Training times, which is estimated in term of number of iteration, are compared. The resulting numbers of support vectors are compared. Using RBF kernel, all three tools finished in short number of iteration but produce consistently large number of support vectors. While using polynomial kernel, significantly higher number of iterations are needed. Using polynomial kernel, training was slowest in SVMLight while number of support vectors is reasonably

low. All three tools generate comparable number of support vectors for the same kernel even with different C values.

6.3.2 Comparing Number of Support Vectors

Comparative training results on all datasets using all three tools are provided in Table 6.10. All training exceeding a certain threshold of iteration and time using certain kernel and hyper parameters has been skipped and the results are entered in the table as N/A. Only LIBSVM was trained for both binary and multiclass as SVMLight and SVMTorch that we used only implements two-class SVM. All tools generate reasonably the same number of support vectors when using RBF kernels.

Table 6.10 Training Result (number of Support Vectors)

SVM Package	Kernel	C	Data Sets				
			WBC	CHD	TTT	VT	HWD
LIBSVM	RBF	1	569	297	627	112	3811
		10	569	297	423	76	3810
	Polynomial	1	32	173	626	126	728
		10	32	173	422	74	728
SVM Light	RBF	1	569	N/A	949	305	N/A
		10	569	N/A	949	306	N/A
	Polynomial	1	32	N/A	328	59	N/A
		10	32	N/A	848	58	N/A
SVM Torch	RBF	1	561	283	N/A	N/A	N/A
		10	561	279	N/A	N/A	N/A
	Polynomial	1	N/A	N/A	N/A	N/A	N/A
		10	N/A	N/A	N/A	N/A	N/A

Note: WBC, TTT and VT are 2 class problems; CHD and HWD are multiclass problems. SVMLight and SVMTorch are 2 class packages.

6.3.3 Comparing Training and Test Accuracies

Table 6.11 gives the comparison of the training accuracy for each package. RBF kernel and Polynomial kernel of degree 2 were used. The values of C = 10 are used in all cases. LIBSVM produce on average the highest training accuracy.

Table 6.11 Training Accuracy (in %)

SVM Package	Kernel	C	Data Sets				
			WBC	CHD	TTT	VT	HWD
LIBSVM	RBF		100	100	92.80	95.86	100
		10	100	100	98.33	98.85	100
	Polynomial	1	97.54	78.79	85.70	95.17	100
		10	97.36	80.47	90.71	97.93	100
SVMLight	RBF	1	100	N/A	100	95.63	N/A
		10	100	N/A	100	96.32	N/A
	Polynomial	1	97.89	N/A	98.33	99.77	N/A
		10	97.89	N/A	97.51	100	N/A
SVMTorch	RBF	1	100	95.63	N/A	N/A	100
		10	100	100	N/A	N/A	100
	Polynomial	1	N/A	N/A	N/A	N/A	100
		10	N/A	N/A	N/A	N/A	100

A comparison of the test accuracy for the three tools was also made. Table 6.12 gives the test accuracy for 10-fold cross-validation training using SVM with RBF kernel.

Table 6.12 Summary of Test Accuracy (in %)

SVM Package	Data Sets				
	WBC	CHD	TTT	VT	HWD
LIBSVM	96.13	58.25	70.67	93.79	97.91
SVMLight	96.24	N/A	70.67	93.79	N/A
SVMTorch	95.43	N/A	N/A	N/A	96.54

We finally selected LIBSVM as a base SVM package in our test of SVM for character recognition as well as in the hybrid SVM/HMM online word recognition system. The major reasons are; first ease of integration into the system where LIBSVM is an already multiclass solution while the other two are not; second it is better in training and test accuracies.

6.4 Character Recognition Using SVM

After these preliminary experiments which allow to select the SVM package, more in-depth experiments were conducted to investigate the usage of SVM in online character recognition. In these experiments, we use the IRONOFF database, UNIPEN database and a combination of the two databases together called IRONOFF-UNIPEN. The description and details of each database are given in Section 6.2 and Table 6.2.

6.4.1 Experiments on SVM for Character Recognition

For the experiments, a feature extractor module extracts the 7 local features for each point of the online signal in the example character (see section 5.2.3). These 7 features are chosen since they are simple to obtain and have been used by Poisson (Poisson, 2002) in other similar experiments using TDNN and MLP NN. Therefore, for each example character there are 210 feature values which are the inputs of the SVM. For our character recognition, we use LIBSVM library with RBF kernel, since RBF kernel has been shown to give better recognition result. Grid search on a 10-fold cross validation were performed on the databases in order to choose the best values for the C and γ parameters for the final SVM training.

It is observed that C values between 2 and 8 and gamma values between 2^{-7} and 2^{-5} yielded the best character recognition rate. We have chosen a single pair of C = 8 and gamma = 2^{-5} for our training on all databases since the results obtained shows that individual grid search on the datasets yields almost similar C and gamma values for majority of the datasets.

Table 6.13 shows recognition performance for the recognizer using IRONOFF-UNIPEN database. The table gives the total number of examples in the training and test set as well as the accuracy, number of support vectors and the training time taken. The training time and the number of support vectors seem to be proportional to the size of the training data, which is normally the case. The bigger the training size, the longer is the training time and the bigger the number of support vectors.

However, test accuracy for each dataset does not follow the normal case where a bigger dataset used in training gives better accuracy. This is due to the quality of the character in the different dataset. A digit, in particular, is normally written consistently uniformly if compared to lowercase or uppercase letter making them less varied thus less confusion during recognition.

Table 6.13 Detail Recognition performance of SVM on IRONOFF-UNIPEN character database

Data Set	Training Set	Test Set	Test set accuracy (%)	nSV	Training time (s)
Digit	13451	6270	98.68	3014	497
Lowercase	42778	20172	93.76	15696	5897
Uppercase	25662	11621	95.13	10035	2808

When compared with two other character recognizers based on neural networks; the MLP and TDNN, our SVM based character recognizer consistently performs better. Table 6.15 shows the comparison between MLP NN, TDNN and SVM character recognizers, trained and tested on IRONOFF and UNIPEN databases.

Table 6.14 Comparing recognition performance between TDNN and SVM for IRONOFF and UNIPEN databases

	IRONOFF database			UNIPEN database		
Data Set	MLP	TDNN	SVM	MLP	TDNN	SVM
Digit	98.2	98.4	98.83	97.5	97.9	98.33
Lowercase	90.2	90.7	92.47	92.0	92.8	94.03
Uppercase	93.6	94.2	95.46	92.8	93.5	94.81

As can be observed, the recognition rate or the accuracy using SVM is better than TDNN and MLPNN for all datasets in the two databases. These are due to the effectiveness of maximal margin optimization and structural risk minimization (SRM) approach to learning used in SVM. NN which normally uses empirical risk minimization (ERM) does not give an optimal classifier since there can be many classifiers obtained given an initial set of parameters to start with. Better accuracies for SVM are also obtained for IRONOFF-UNIPEN database as seen in Table 6.15.

However, a fair comparison need not be just by looking at the recognition accuracy. A practical recognizer should be small in size, carrying as small as possible number of parameters. The free parameter columns for MLP, TDNN and nSV column for SVM in Table 6.15 gives a comparison for this. In such case, TDNN recognizer is a clear winner because the total number of free parameters is small. This is due to the weight sharing scheme within the structure of the TDNN. According to the table, SVM seems to have a small number of parameters indicated in the nSV column, which is the total number of support vectors.

Table 6.15 Comparing recognition performance and number of parameters using MLP, TDNN and SVM for IRONOFF-UNIPEN database

	MLP		TDNN		SVM	
Data Set	**Free par.**	**Rec Rate(%)**	**Free par.**	**Rec Rate(%)**	**nSV**	**Rec Rate(%)**
Digit	36110	97.9	3790	98.4	3014	98.68
Lowercase	37726	91.3	8926	92.7	15696	93.76
Uppercase	37726	93.0	8926	94.5	10035	95.13

Since a support vector is actually an example from the training set, its size is actually a multiple of the dimension of the feature vector representing the example. This can be large. In our SVM, this is 30 x 7 or 210. To fairly compare the number of parameters for each MLP, TDNN and SVM, let's take the parameters for digit recognizer. For MLP, it is 36,110, for TDNN 3,790, but for SVM, it is 3,014 x 210 = 632,940 or 18 times larger than MLP NN. One way to tackle this large model is to use compression. Parameters are stored in compressed form. During recognition, the model will be expanded dynamically as required.

As discussed in section 4.4.3, SVM can be made to give posterior probability outputs. Since we will eventually use the SVM in the word recognizer, we decided to train and test the SVMs for handling probabilistic output using IRONOFF and UNIPEN databases. In the training, the SVM was trained with the option for probabilistic output and recognition were then done using the model for probabilistic

output which gives probability values for each class as the outputs. The correct recognition is the character class which gives the highest probability value. The recognition accuracy does not differ much between using SVM with probabilistic model or non probabilistic model, as seen in Table 6.16.

Table 6.16 SVM distance vs. probabilistic SVM based recognition for IRONOFF and UNIPEN Databases

	IRONOFF database		UNIPEN database	
Data Set	**SVM**	**SVM prob.**	**SVM**	**SVM prob.**
Digit	98.83	98.68	98.33	98.35
Lowercase	92.47	92.42	94.03	94.14
Uppercase	95.46	95.45	94.81	94.85

6.4.2 Character Recognition Summary

In all the experiments, the results have shown that the recognition rates of characters using SVM character recognizer are significantly better than other methods compared, due to structural risk minimization implemented by maximizing margin of separation in the decision function. However, the increase in recognition rate is not without some impact. SVM model size is characterized by the number of support vectors obtained in the training. Storing these support vectors for recognition requires larger memory as compared to NN weights since each support vector is a multidimensional feature vector. The number of support vectors can be reduced by selecting better C and gamma parameter values through a finer grid search and by reduced set selection (Burges, 1996) (Downs, 2001). The comparison of recognition results of SVM with probabilistic output and SVM distance output shows that both are comparable.

In section 6.6, we present the work on integrating the SVM character recognizer into the HMM based word recognition framework. However, we first describe some other works which makes use of SVM that the author has undertaken in the following section.

6.5 Experiences in Implementation of SVM in Other Areas

This section reports some results of the author's work in using SVM for applications in other areas. It provide some information regarding the effectiveness of SVM in two areas; first, an area very close to online handwriting recognition, which is on online handwritten mathematical expressions recognition and second; an area which is not directly related to handwriting recognition. The first result is on comparison of the usage of SVM and TDNN in mathematical expressions recognition which is part of a research (Awal, 2008) within the same laboratory. The second work (Ahmad, 2007) was performed for the state owned Malaysian power producer, Tenaga Nasional Berhad (TNB) on customer fraud prediction using SVM.

6.5.1 SVM in Mathematical Expressions Recognition

In a larger perspective, a framework for online handwritten mathematical expression recognition was proposed. The architecture aims at handling mathematical expression recognition as a simultaneous optimization of symbol segmentation, symbol recognition, and 2D structure recognition under a mathematical expression grammar. Its components are hypothesis generator that performs a 2D grouping of elementary strokes, a classifier that labels the hypothesis according to a predefined set of symbols, a cost function defining the global likelihood of a solution, and a dynamic programming scheme that gives the best global solution. For recognizing the elementary strokes, TDNN has been chosen to be the base recognizer. A TDNN was trained using large datasets of online handwritten mathematical symbols that has been collected. The database consists of Greek symbols, elastic symbols, arrows, functions as well as capital letters, small letters and digits. SVM has been used to compare the recognition results with TDNN using the same collected database.

Table 6.17 Comparison of TDNN and SVM on isolated Mathematical symbol recognition

Symbols	Number of Classes	Training set	Testing set	Recognition accuracy (%)	
				TDNN	SVM
All Symbols	223	40128	22294	60.21	79.42
Greek symbols	30	5400	3000	72.7	85.88
Elastic	37	6660	3698	78.88	89.13
Arrows	59	10619	5900	79.95	92.51
Functions	35	6298	3500	73.34	87.71
Capital letters	26	4680	2600	77.42	94.85
Small letters	26	4674	2600	74.5	92.00
Digits	10	2719	1367	87.5	96.60

As can be seen in Table 6.17 and Figure 6.2 , SVM gives better recognition accuracy for all types of symbol. These are achieved by a simple 3-fold cross-validation procedure in the selection of SVM parameters and the penalty term. It shows that SVM is a robust classifier which can be adapted for use in any handwriting related recognition.

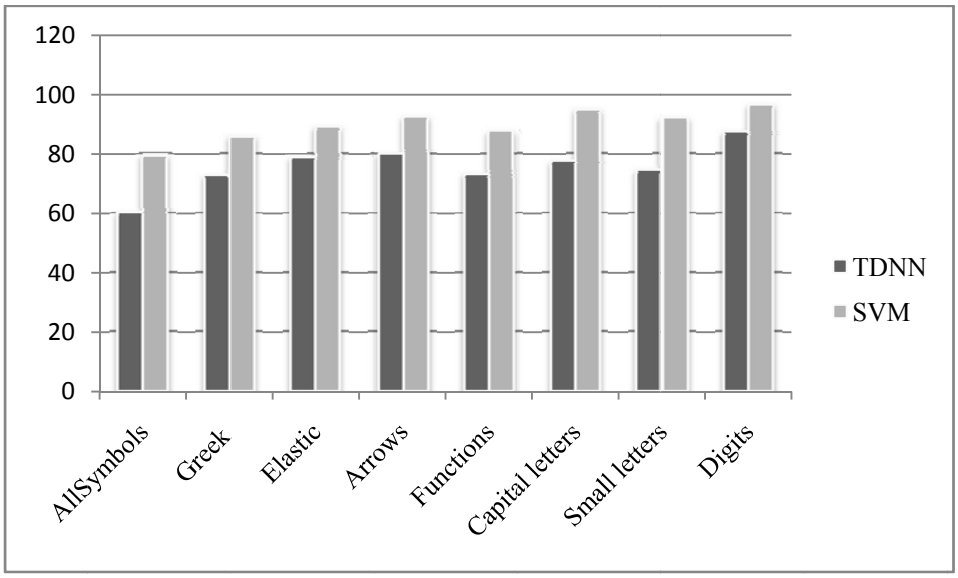


Figure 6.2 Comparison of TDNN and SVM on isolated symbol recognition

6.5.2 SVM in Electricity Fraud Prediction

In this second related work on SVM, we developed an intelligent system to detect fraudulent customers for a Malaysian power company, Tenaga Nasional Berhad (TNB). The aim is to create a list of fraudulent customers from the company customer database using SVM so that instead of spending a lot of money on inspection campaign on customers, the list of likely fraudulent customers will be generated by using an SVM predictor trained using samples of verified customers.

Of the 0.4 million customers of TNB from a particular state, an estimated 7.5% of the total customers have been checked by visiting selected customer premises. Out of that, only 6% have been confirmed fraudulent (the strike rate). With the use of the SVM based predictor, it was hoped that a predicted fraud list will give a high strike rate much better than the manual method of 6%, giving much cost saving from not having to perform manual inspection on large customer base. To train the SVM, data from 13,000 customers that have been checked are used. Features used are the electricity consumptions for the last 85 months resulting in an 85 dimensional feature data. Missing data are projected by using the average profile of all customers. For SVM parameter selection, 10-fold cross validation was used. In validating the predictor, we used a few different data sets.

Table 6.18 Fraud prediction Accuray

Dataset	**10-fold cross validation accuracy**
332 customers initial set	76.51
190 customers verified set	93.12
2000 customers verified set	73.4
13000 customers verified set	68.56

Table 6.18 gives a summary of the results obtained in the course of training and validation of the SVM predictor. The table gives the validation accuracy using 10-fold cross validation for the different sample data that was used. A conclusion that can be made is that the SVM fraud predictor has been able to predict correctly the

fraudulent customers for more than 60% of the time which is more than 10 times better in accuracy than the strike rate of 6% by manual method.

6.6 Word recognition Using Hybrid SVM/HMM

The hybrid word recognition system that we developed was evaluated by conducting some experiments using the IRONOFF word database. We trained and tested the system on each of the databases in IRONOFF; the cheque word, English word, French word and the overall word databases separately. Cheque words database contains a lexicon of 30 words, while English contains 26 words, French 171 words and the overall words altogether contains 197 words. The aim of the tests was to investigate if our method of preprocessing, feature extraction, segmentation and SVM training are suitable and gives good recognition results and with that we will make recommendations for the implementation of such hybrid system.

First, initial Character SVMs were trained and tested for each databases before they are used in each baseline word recognition systems. Segmentation of words into characters for the baseline system was done by a commercial recognizer, guided by the actual label during recognition. The use of actual label is supposed to help the commercial recognizer during its recognition and thus gives a very good recognition for segmentation to be done perfectly, resulting in a good character database. With this we hope to start off with a good SVM. As discussed in section 5.3.6, the process of training the system is done in a few cycles of resegmentation of word databases and retraining of character SVMs until there are no more improvements in the performance of the character SVMs.

In the following subsections, we first discuss a simple example of the processes involved in all the steps for the word recognition and resegmentation for a simple word “hi”. The example demonstrate the processes involved in the hybrid system, showing the character recognition functionality of SVM and the dynamic programming functionality of the HMM in calculating the word probability for each word in the lexicon. For overall evaluation of the hybrid word recognition system, we discuss the results obtained during the overall process of the SVM training at the

character level and the recognition at word level and subsequently segmentation and retraining of the character SVMs for all the word databases. We then analyze the errors caused and discuss issues related to them.

6.6.1 A Word Recognition Example

For our word recognition system training, we demonstrate here the processes involved in the recognition and resegmentation of a simple example word consisting of two characters, the word “hi”. For this demonstration, the word “hi” have been added into the lexicon of English words and a character SVM trained with characters from the English words in the IRONOFF database was used. As can be seen in Figure 6.8, characters ‘h’ and ‘i’ are part of the characters in the words of the English word database. The signal for the word “hi” is as shown in Figure 6.3. To make the analysis simple, in writing the word “hi”, the dot for the character ‘i’ was not written, so that the word “hi” is a word with only one stroke.

The portions of the original signal and the pre-processed signal are shown in (a) and (b) respectively. It shows the change in coordinates after the preprocessing and normalization. For details of the preprocessing that we performed, refer to Section 5.2.2 on preprocessing and normalization.

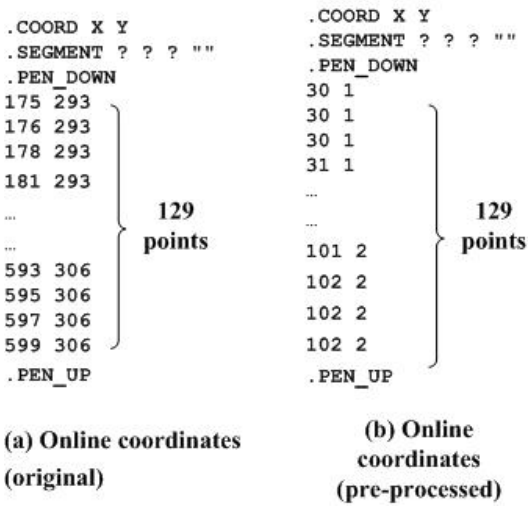


Figure 6.3 The online signals of the word "hi"

The word “hi” contains only one stroke with 129 points. The already pre-processed word signal is segmented into slices by a very simple algorithm which tracks each point from the beginning of the stroke, and groups them into slices from minimum to maximum points or minimum to maximum points repeatedly over the stroke. In our example here, the segmentation into slices of the word “hi” yields 6 slices as shown in Figure 6.4. Once we have the slices, they are combined to form character hypothesis, which can contain the minimum of 1 slice up to a maximum number of ω slices.

(a) Word signal – “hi”

(b) The 6 slices

Figure 6.4 The 6 Slices from the word "hi"

In generating the hypothesis, the value for ω was chosen to be 5 for this example but a value of 7 was used in the rest of the system, as explained in section 5.3.4 and also by experimental results. The value of ω chosen affects the recognition accuracy and the time taken to perform the recognition. A large value of ω results in larger number of hypothesis and longer recognition time but with higher possibility for correct recognition. These character hypotheses are preprocessed, resampled and recognized by the SVM character recognizer which gives the probabilities for the hypothesis to belong to any one of the character classes. For each word in the lexicon, the dynamic programming step is used to find best path combining the hypothesis giving the best score for generating the particular word in the lexicon.

In this example, using 6 slices, there are 20 character hypothesis possible and can be verified by the formula given on page 116. The character hypotheses are then resampled to a uniform 30 points and 7 local features are extracted from each point, resulting in a 210 dimensional feature vectors. Each set of feature values are then passed to the character SVM to yield the probability array which keeps the

probabilities of character classes for all hypotheses. This probability array is kept in memory during the word score calculation for each word in the lexicon. Keeping it in memory reduces the computation time as character probabilities needed for each calculation are made available in the matrix and SVM need not be called every time to get their values.

We show in Figure 6.5 the trellis which was constructed in order to evaluate the score for the lexicon word “hi” itself. The trellis to be constructed for the evaluation of the score for each word in the lexicon but for simplicity sake we show only this one trellis which happens to be the word to be recognized and segmented.

Figure 6.5 matches the word recognition graph of Figure 5.16 except that it is rotated 90 degrees clockwise. In the trellis, each cell represents the probability of the hypothesis that combines the slice(s) from slice (t – q + 1) to slice t where t is the slice number and q is the number of slices. For example, the circled value -0.13 represents the probability that a hypothesis containing 4 slices - slice 0…3, is character ‘h’ and the value -0.04 to the left of -0.13 represents the probability that a hypothesis containing 3 slices – slice 1…3, is character ‘h’. This a constrained graph where for the first character, only hypothesis starting with slice 0 is valid and from one character hypothesis, the path lead to the next character hypothesis starting from the next slice after the last slice in the current hypothesis. The values indicated by – INF are undefined since we cannot have a hypothesis with such slices, for example, it is not possible to have a hypothesis that have 4 slices ending with slice 2.

In our example here, the best path for the word “hi” is as indicated by the arrow; for character ‘h’, it’s the hypothesis containing 4 slices – slice 0…4 having a log probability score of -0.13 and for character ‘i’, it is the hypothesis containing 2 slices – slice 4 and 5 with a probability score of -0.21, giving total word probability score of -0.34. The final normalized word score which is taken for comparison against other word lexicon is the total word probability divided by length of the word ‘hi’ giving the score of -0.17. For other words in the lexicon, the same thing is done to obtain the final normalized word score. The word with the highest normalized word score is chosen as the recognized word. The verbose output of the

recognition and segmentation process for the word “hi” using the English word lexicon is shown in Appendix C.

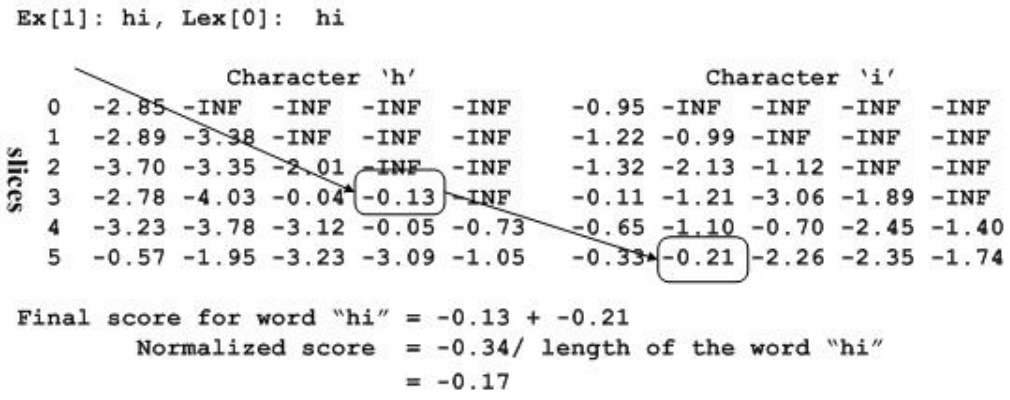


Figure 6.5 Trellis for probability score of each hypothesis and the best path for scoring the lexicon word “hi” itself.

As can be seen in appendix C, the score for some of the word lexicon are –INF. The value of –INF are assigned in the cases where the number of characters in a lexicon word is bigger than the number of slices cut from the word to be recognized, since there is not enough slice(s) to sufficiently represent each character. For our simple example given here, the highest word score (at position 1) is -0.17 which is for the lexicon word “hi” itself, meaning that our recognizer recognizes the word correctly. The correct segmentation points for the word are such that for letter ‘h’, it consists of slice 0 to 3 and for letter ‘i’, it consists of slice 4 and 5 as seen in Figure 6.6. Information about these correct segmentations is used to regenerate characters, hopefully better segmented ones. In the overall training which involves word databases, character databases can be generated for retraining of SVM.

Character 'h' consisting of slices 0, …, 3

Character 'i' consisting of slices 4 and 5

Figure 6.6 Character Segmentation for the word "hi"

6.6.2 Comparing Word Recognition Performance

When we want to evaluate our hybrid word recognition system, we need to use the complete word database to measure the percentage of correctly recognized words from the database (those words having true class in position 1). However, other reasonable comparison of the performance would be to compare the percentage position of the true class in certain top position other than 1, between 2 to n. For example, we can take a word as "correctly recognized" if the true class is in the top 3 position and compare recognition performances based on this. Other than that, the average position of the true class can also be used.

In summary, the measures that can be used to evaluate the performance of a word recognizer are as follows:

a) Top(n) for n = 1, …, N : the percentage of samples for which the true class are in the top n position of the candidate list. For example, Top(3) performance measure of 95% means that 95% of the samples words tested have the true class to be among the top 3 positions.

b) $\overline{pos}$: the average position of the true class in the candidate list generated by the recognizer. The value of 1.0 is the best, which can only be achieved if all the test data are correctly recognized.

In the results for word recognition comparison for each training cycles which we report the results in section 6.6.5, we mainly used the Top(n) measures for n from 1 to 10.

6.6.3 Character Database Generation.

To bootstrap our system with a properly trained SVM, we trained character SVMs using characters segmented from the word database. As mentioned, each word in the word database was segmented using an API library of a commercial word recognizer to obtain database of characters. The size of the character databases generated (training set and testing set) for cheque words, English words, French words and all words databases are as in Table 6.19. The relative distribution of characters generated is shown in the charts of Figure 6.7, Figure 6.8 and Figure 6.9.

Table 6.19 Number of characters in generated character database

	Number of characters	
Word Database	**Training set**	**Test set**
Cheque	39578	19675
English	11270	5523
French	107201	53634
All Words	158049	78832

As we can observe from the figures the database of cheque words characters only contain 21 of the lower case characters and also without the upper case and French accented characters. The database of English word characters contains a fair number of upper and lower case characters. In the French words character database, all character sets in the character lexicon are used. However, the number of examples of upper case characters is very low compared to the lower case characters. The number examples of French characters are also considerably adequate. This class imbalance in data sets affects the training of SVM.

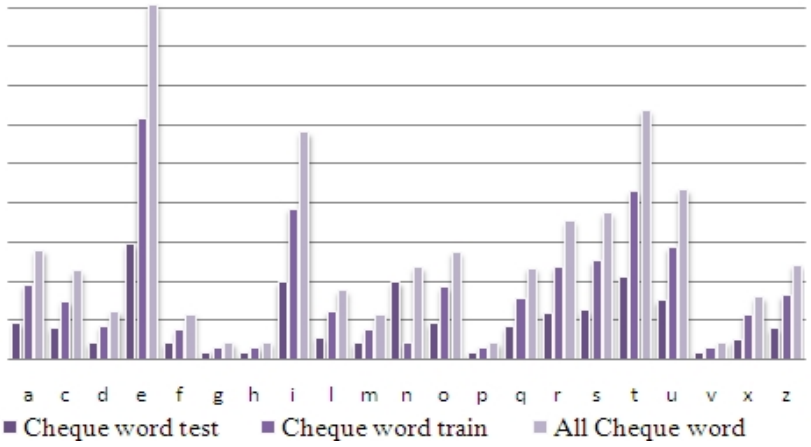


Figure 6.7 Distribution of characters in the generated cheque word character database. Only a subset of lower case characters are present.

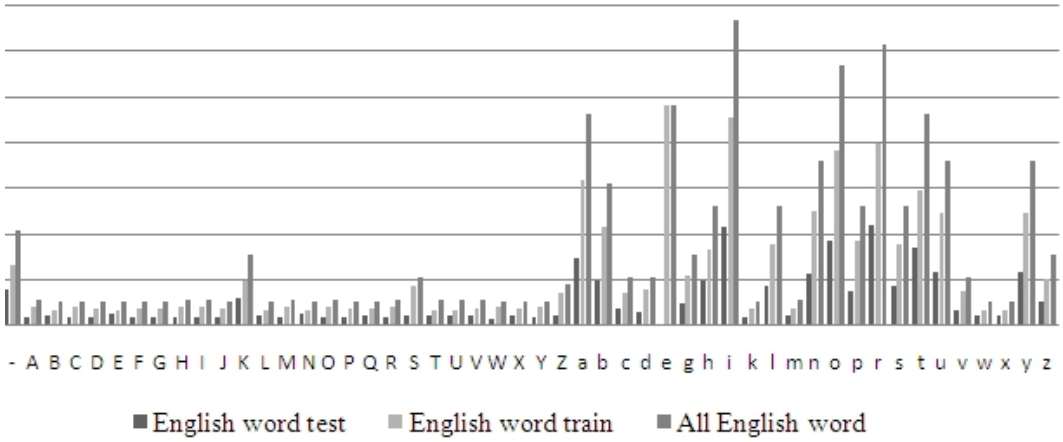


Figure 6.8 Distribution of characters in the generated English word character database. Some character classes from character lexicon are not present.

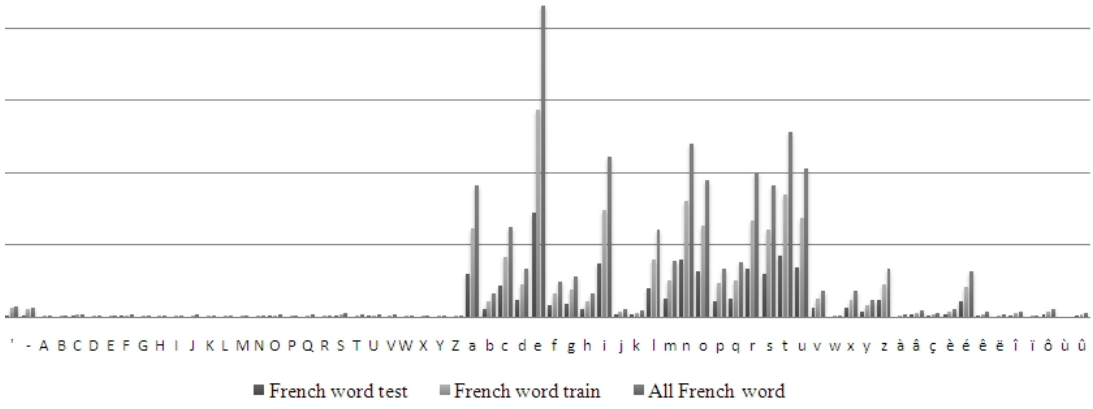


Figure 6.9 Distribution of characters in the generated french word character database. All character classes in the character lexicon are present.

Segmentations were done only on correctly recognized words. For comparison purposes, we noted the recognition accuracy of the commercial recognizer in the process of doing the segmentation.

Table 6.20 Word Recognition accuracy of during segmentation

	Recognition accuracy (%)	
Word Database	**Training set**	**Test set**
Cheque	96.76	96.03
English	99.72	99.44
French	92.00	94.33
All Words	96.16	96.60

Table 6.20 and Figure 1.1 show the word recognition accuracy obtained by the commercial recognizer. The recognizer was built using a hybrid of ANN and HMM. Although the recognition process for the purpose of segmentation were guided by the label, we did not get perfect recognition and thus the characters generated does not come from the entire words databases since only correctly recognized words contribute the characters.

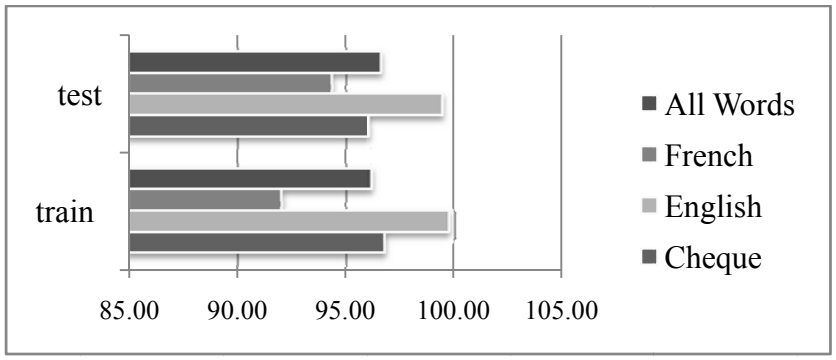


Figure 6.10 Recognition accuracy during segmentation

6.6.4 Training of Character SVMs

The character databases obtained from section 6.6.3 are then used to train our character SVMs. We trained the SVMs separately for cheque words, English words and French words, as well as all three together as the overall word database. For

selecting the best parameter values (γ and C) to train a final SVM, a 10 fold cross validation was done on the cheque word character database. We obtained the RBF kernel parameter values of $\gamma = 0.03125$ and $C = 2$ to be optimum for training the SVMs. The same parameter values were used for training the other SVMs. Each SVM was trained using the characters from the training set and tested using characters from a separate test set indicated in Table 6.20.

Table 6.21 Performance of the character SVMs

Character Database	Recognition rate (%)	Number of Support Vectors
Cheque	85.47	20,709
English	80.46	8,591
French	81.66	43,800
All words	84.33	86,347

We can make an observation here that the recognition rate of the trained SVM is not very high. This is not that important however, as the overall word recognition relies on combination of a number of SVM outputs where a low probability for a character in the word can be compensated by high probability in the other characters in the word. This is evident in the word recognition results in the following section, for English words. Although the character SVM for English words gives the lowest recognition rate among them, word recognition rate for English words is the highest among them (see Table 6.22). One thing to note however is the significant size of the number of support vectors. Number of support vectors in each SVM is as large as 50% of the total number of example characters in the generated character databases. This can be due to the values of the training parameters chosen. As understood, bigger number of support vectors means longer recognition time since each support vector is involved in the calculation of the output.

6.6.5 Recognition Result for Baseline Word Recognition System

The baseline system is the word recognizer using the character SVM trained with the characters segmented from the word with the commercial recognizer. We

performed recognition on the individual word databases and regenerate new character database using the segmentation points generated during the recognition.

Table 6.22 Word recognition rates of base recognizer

Database	Lexicon size	Recognition rate (%)			
		Top(1)	Top(2)	Top(3)	Top(10)
English	26	98.77%	99.44%	99.50%	100%
Cheque	30	76.71%	91.64%	95.71%	99.99%
French	171	63.25%	77.90%	85.15%	98.86%
All words	197	64.53%	79.05%	86.20%	99.91%

We obtained the word recognition rates as shown in Table 6.22 and graphically represented in Figure 6.11. The result shows that Top(1) recognition rate is not very high, where all except English word databases gives below 98% recognition rate. However, the Top(10) recognition rate of almost or above 99%, indicates that although the recognizers made some errors in word recognition, they are still within the Top(10) positions. The analysis of the errors that occur during word recognition will be discussed in section 6.6.9.

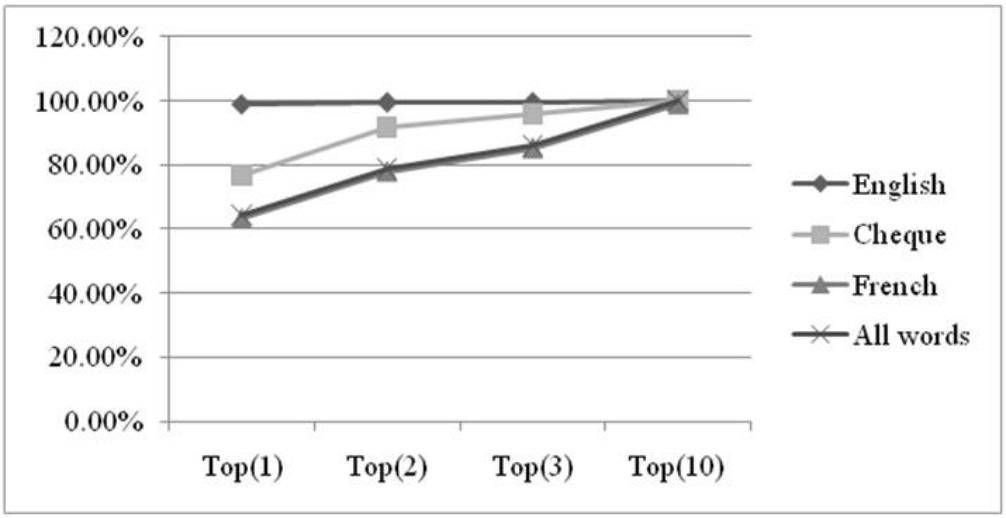


Figure 6.11 Word recognition rates for base recognizer

6.6.6 Retraining of SVMs

To demonstrate further effectiveness of the system, we retrain the individual character SVM for the English word database. The character SVMs recognition rate, the number of support vectors for each character SVM and the word recognition rate of the word recognizer based on the new character SVM are given in Table 6.23.

Table 6.23 Improvements in Character and word recognizer for the English Words

Iteration	Character SVM Rec. Rate (%)	nSV	Word Rec.Rate Top(1)
Baseline	80.46	8591	98.77%
First iteration	74.37	8449	98.49%
Second iteration	74.27	8296	98.83%
Third iteration	74.32	8349	98.99%

As observed from the table, the performance of the new character SVM were less than the baseline character recognizer but it did not change the word recognition rate too much. The word recognition rates were within 1% above and below the baseline word recognition rate. The number of support vectors is also generally lower than the baseline character recognizer which means the SVM model is getting smaller in size. As we performed word recognition and resegmentation repeatedly, the word recognition rate converges to around 98.9%.

6.6.7 Incorporation of Junk Characters in Retraining of SVMs

From Table 6.23, it can be observed that the SVM training performance did not actually improve after the many iterations and in turn does not improve very much the word recognition. The SVM character recognizer may have been presented with many of the “characters” it never have seen which are character hypotheses that does not look like any of the characters in the character classes. Worse still the hypothesis might look like a real character but is not the actual character of the word in the lexicon, for example, a part of the character ‘d’ can look like character ‘c’. We call the non characters as junk characters. These junk characters can contain part of a

character or combination of a few characters. In some cases it can give high score to a word in the lexicon which is not the true word. It might be a good idea to create a class for the SVM that represents these junk character hypotheses. Junk examples can be created from the word signals and added to the character examples generated by the previous training stage. The combined database of characters and junk examples can then be used to retrain the SVM which have an extra class called junk.

To select junk examples, the following guide can be used:

(a) To only select a very small and relevant number of junks, we can select only one character hypothesis at each training stage that cause recognition error as junk example for each training word.

(b) For choosing the best junk example, we can compare the character hypotheses that made up the true word label (true word) and the hypotheses that made up the word with the best score (best word). A value, say γ that represents the probability that a given hypothesis is a true character hypothesis can be calculated for each hypothesis. A hypothesis that has their two γ's from the true and best word differs the least, is taken as the junk character example.

This idea of using junk class has been described and used by (Tay, 2002) in his work on hybrid NN/HMM for offline word recognition. We contemplate in using a junk class in our SVM; however, the idea was put off. In our work, since we do not use ligatures to represent concatenations between character hypotheses, we felt that it is not necessary to use a junk class. Our word HMMs are concatenations of character HMMs without the use of ligatures. In the work of (Tay, 2002), which uses NN, junk class and ligature class may compete as a ligature can be taken as a junk class. Furthermore the junk is only used to flatten out other character probabilities when a hypothesis is not a character. For our work, SVM probability outputs are representative enough for each character class.

6.6.8 Result Comparisons with Hybrid of TDNN and HMM approach.

A comparison can be made of the results obtained earlier using a hybrid of SVM and HMM with the results obtained using a global word training using TDNN and HMM by Caillault (2005). Table 6.24 shows the recognition results obtained. In the table, the results were shown for TDNN with 1 state (état, in French), 2 states and 3 states and four different training criteria; either maximum likelihood training (ML) or maximum mutual information (MMI) and a combination of MMI and ML and MMI, ML and TDNN.

Table 6.24 Recognition result Using TDNN for IRONOFF word

Critères	ML	MMIs	MMIs-ML	MMIs-ML-TDNN
1 état	77,43	83,82	86,34	87,09
2 états	80,87	87.46	88.22	90,51
3 états	84,69	90,57	92,01	92,78

Comparing Table 6.22 and Table 6.24, it can be observed that the recognition rate using hybrid of SVM and HMM that we use gives a recognition rate of 64.53% for Top(1), 79.05% for Top(2) and 86.20% for Top(3). For the hybrid of TDNN and HMM, the recognition rate is from as low as 77.43% using ML estimation and 1-state TDNN, reaching as high as around 92.78% using combination of MMI-ML-TDNN estimation. As with most globally trained, or another word, a hybrid system trained at word level, recognition rate is always higher because the output at word level provides correction information to the character level.

6.6.9 Analysis of Errors

We analyze the errors made by our word recognition system on each of the word database. We have divided the errors into minor and major error. Minor error refers to an error that does not result in the first recognized position but still are in the other

two top 3 positions – either 2nd or 3rd. Major error causes the recognition position to be outside the top 10 positions. Among the cause of errors that we observed are due to (a) errors in preprocessing or segmentation, (b) errors caused by wrong detection of the reference lines, (c) wrong word labels which differ from the actual word signal, (d) bad handwriting.

For discussion of recognition errors, we use the English word examples. For the English word database, the words are correctly recognized within the Top(10) position with more than 99% within the Top(3) position. This is evident from the output summary of the recognition process. This may be due to the fact that English alphabets are without accented characters and can be well represented by the classes in the English character SVM as compared to French characters. Our preprocessing does not include taking into account those special accented characters and the presence of the extra stroke(s) for the accented characters which vary in the time they are written can affect the character feature representation.

A few minor and major errors that we analyzed from the English word recognition results are as follows:

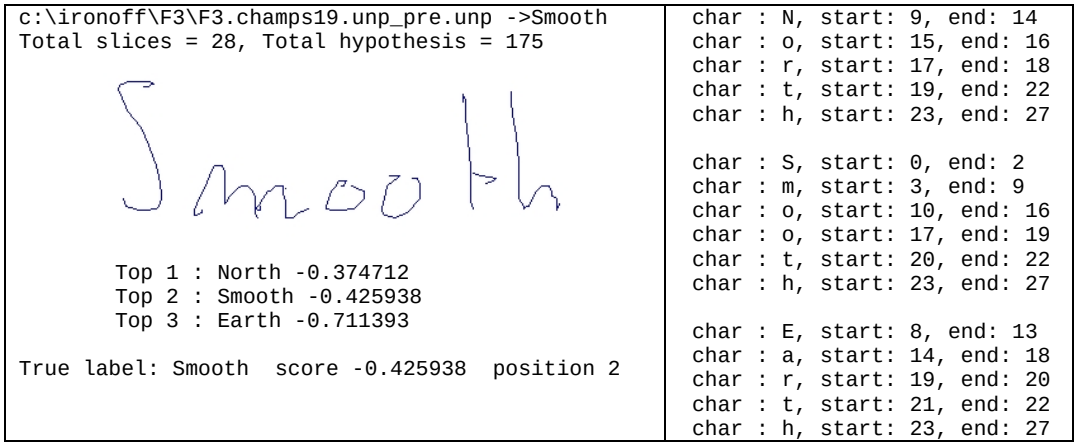


Figure 6.12 Example error: reference line detection

For the example of Figure 6.12, the error in word recognition is caused by an error in the reference line detection. In the calculation of the cost of the best path for word lexicon “North”, the probability for starting character N at slice 9 is higher compared with the hypotheses combination that start from slice 0. This give the

word score for “North” to be higher than “Smooth”. Another observation that can be made is that all top three words end with the letters “th” but the ‘t’’s are represented by hypothesis from different slice combinations. The case of example in Figure 6.13 is similar where the word with the top score starts with hypothesis that do not contain slice 0. This again is due to a high probability of character B for that hypothesis against earlier hypothesis.

```
c:\ironoff\F55\F55.champs11.unp_pre.unp ->Ku-
Klux-Klan
Total slices = 36, Total hypothesis = 231
```

```
       Top 1 : Between -0.787303
       Top 2 : Voice -0.839355
       Top 3 : Ku-Klux-Klan -0.874788

True label: Ku-Klux-Klan  score -0.874788
position 3
```

```
char : B, start: 8, end: 13
char : e, start: 14, end: 16
char : t, start: 17, end: 17
char : w, start: 18, end: 24
char : e, start: 25, end: 29
char : e, start: 30, end: 32
char : n, start: 33, end: 35

char : V, start: 16, end: 22
char : o, start: 23, end: 29
char : i, start: 30, end: 30
char : c, start: 31, end: 31
char : e, start: 32, end: 35

char : K, start: 0, end: 3
char : u, start: 4, end: 6
char : -, start: 7, end: 7
char : K, start: 8, end: 12
char : l, start: 13, end: 15
char : u, start: 16, end: 19
char : x, start: 20, end: 20
char : -, start: 21, end: 21
char : K, start: 22, end: 26
char : l, start: 27, end: 28
char : a, start: 29, end: 32
char : n, start: 33, end: 35
```

Figure 6.13 Example error: reference line detection

Another example of error that we observed is an incorrect recognition caused by incorrect labeling. In the example shown in Figure 6.14, the image of the signal is clearly “North” but the label is “Job”. The Top(1) position correctly belongs to the word “North” with the highest score.

```
Ex[1382]:
c:\ironoff\F81\F81.champs10.unp_pre.unp ->Job
Total slices = 18, Total hypothesis = 105
```

```
       Top 1 : North -0.198316
       Top 2 : Earth -0.576795
       Top 3 : Money -0.776496

True label: Job  score -1.221589  position 8
```

```
char : N, start: 0, end: 3
char : o, start: 4, end: 7
char : r, start: 8, end: 9
char : t, start: 10, end: 13
char : h, start: 14, end: 17

char : E, start: 0, end: 5
char : a, start: 6, end: 8
char : r, start: 9, end: 9
char : t, start: 10, end: 13
char : h, start: 14, end: 17

char : M, start: 0, end: 5
char : o, start: 6, end: 6
char : n, start: 7, end: 8
char : e, start: 9, end: 11
char : y, start: 12, end: 17
```

Figure 6.14 Example error: wrong label.

Another example of error made during preprocessing is shown in Figure 6.15. This error is cause by an error during the preprocessing stage where the spurious signal at the first stroke in the top bar of character J that was not corrected due to our simple preprocessing procedure. This cause the bar to resemble an s. However the Top 1 score is very close to the true word at Top 2 score.

```
Ex[7997]: c:\ironoff\F3\F3.champs10.unp_pre.unp
->Job
Total slices = 9, Total hypothesis = 42
```

```
       Top 1 : son -0.361563
       Top 2 : Job -0.366574
       Top 3 : sous -0.508704
True label: Job  score -0.366574  position 2
```

```
char : s, start: 0, end: 0
char : o, start: 1, end: 3
char : n, start: 4, end: 8

char : J, start: 0, end: 3
char : o, start: 4, end: 5
char : b, start: 6, end: 8

char : s, start: 0, end: 0
char : o, start: 1, end: 3
char : u, start: 4, end: 7
char : s, start: 8, end: 8
```

Figure 6.15 Example error: preprocessing

6.6.10 Conclusion

Our hybrid word recognition system has been implemented with simple preprocessing, segmentation and feature extraction procedures but proven to work quite well, especially for English words due to its simpler character sets and smaller word lexicons. In summary, for building SVM character recognizer, our SVM parameters have been chosen by 10-fold cross validation to give the best recognition when we make use of C parameter of 2 and gamma parameter of the RBF kernel to be 0.03125. For segmentation, we have sliced the word into slices from maximum to minimum or minimum to minimum y-axis points and selected the size of a hypothesis to be containing a maximum of 7 slices. We showed that our algorithm for word score calculation is able to give Top(10) word recognition score of greater than 99% for words in the IRONOFF database.

6.7 Summary

In this chapter we describe the databases that are used in the course of testing the hybrid SVM/HMM word recognition system at the various stages of developments and the results obtained at each stage. We described the databases used for the evaluation and selection of the SVM tool, databases for the testing the character SVMs and databases for testing word recognition by our hybrid word recognition system. We then give the results in the testing of our SVM recognizer and the hybrid word recognizer and analyze some of the errors that occur in the recognition process. Finally, we made conclusions from the experiments and the results obtained.

CHAPTER 7

CONCLUSIONS AND FUTURE RECOMMENDATIONS

7.1 Dissertation Contributions

As described in section 1.7, the aim of this work is to address the issue of discriminative training in the hybrid handwritten word recognition system. We have investigated the effectiveness of using SVM in character recognition and its use in a hybrid environment of a segmentation based handwritten word recognition system. In the hybrid system, the discriminative property of SVM is exploited in tandem with the class representative property of an HMM.

We have implemented a hybrid SVM/HMM handwritten word recognition system that caters for a medium sized lexicon that handles connected cursive handwritten words. The system is similar in idea with some existing systems based on discrete HMM or a hybrid of HMM and NN. We recognized that the optimization of the HMM/NN based system can either be at the word level or at character level. In word level, both NN and HMM are optimized based on the output at word recognition level. In our system, we only emphasize optimization at the character level, meaning optimizing the character recognizer based on the segmentation done as a result of word recognition. This is due to the fact that conventional SVM training that we use involves quadratic programming optimization on the dual formulation. Correcting gradient does not propagate from the word level to character or sub-character level training of the SVM.

In the course of this thesis work, we made the following contributions:

a) Evaluated various selection and parameterisazion of SVM for use in handwriting recognition problem. We have shown that SVM with RBF kernel are the most suitable for use.

b) Tested SVM on a few major character databases, proving the effect of various parameterizations in improving character recognition.

c) Implemented and tested SVM with posterior probabilistic measures output. Though they are now standard, we have verified its implementation and usage for handwriting recognition.

d) Implemented a simple method for segmentation based on optimum coordinates points and feature extraction of character segments from words.

e) Use of SVM in a hybrid situation with HMM, in particular the dynamic programming aspect of the Viterbi algorithm in the HMM.

f) Compared SVM/HMM hybrid implementation with other hybrid systems in handwriting recognition and in speech recognition.

7.2 Conclusion

The result of recognition of the hybrid HMM/SVM system is not as promising. However, we believe that the work have not been attempted by other researchers and we have proven that it is possible to implement the hybrid of HMM and SVM similar to the speech recognition counterpart. We analyzed the errors and identified a number of issues that we faced during the implementation. We also listed some recommendation in terms of the training and implementation of SVM character recognizers.

7.3 Future Work

SVM has been used in handwriting recognition by other researchers but mostly at the character recognition level. In our thesis, we have also implemented character recognizers using SVM and shown that SVM recognizers are better than NN based recognizers, in particular the TDNN's and the MLP NN's. However, at the word recognition level, NN based hybrid recognizers have the advantage that word level optimization can be done in parallel with character level optimization due to the gradient descent approach to training.

In our work, we have taken the path of separate optimization for the character recognizer based on segmentation by the word recognizer. There is no information from the word recognizer in term of correcting gradients that are used by the character recognizer in its optimization. Our training of SVM has been by way of solving for the Lagrange multipliers introduced in the dual formulation for the large margin classification in SVM. This involved quadratic optimization and is complex, computationally inefficient and does not allow for outside error correcting information in the SVM training.

A training method that is more efficient that alleviates the difficulties associated with operating in the dual problem is desired. (Kowalczyk, 2001) patented a method for a gradient based method for training SVM in 2006. The method executes an iterative process on the training data to determine the parameters of the SVM. The iterative process is executed on the basis of a differentiable form of a primal optimization problem for the SVM parameters. Generation of support vectors can be done by a method with differentiable penalty by direct minimization of the primal problem.

A suggested future work can be to make use of this gradient based SVM training method in a hybrid SVM/HMM based word recognition system where correcting information from the word level can be used in the character SVM training, comparable to the hybrid NN/HMM system. In this case word level discriminant training which is known to give better recognition results can be performed.

REFERENCES

1. AHMAD, A. R., KHALID, M., YUSOF, R., VIARD-GAUDIN, C. (2004a) Online Handwriting Recognition using Support Vector Machine and Hidden Markov Model. International Conference on AI in Engineering and Technology, ICAIET 2004. Sabah, Malaysia.

2. AHMAD, A. R., KHALID, M., VIARD-GAUDIN, C., POISSON, E. (2004b) Online Handwriting Recognition Using Support Vector Machine. IEEE Region 10 Conference TENCON 2004. . Chiangmai Thailand.

3. AHMAD, A. R., MOHAMAD, A. M., ISMAIL, F.I., ABDUL RAZAK, F. (2007) Intelligent System for Detection of Abnormalities and Probable Fraud by Metered Customers. International Conference on Electricity Distribution, CIRED2007. Vienna, Austria.

4. ALLWEIN, E., SCHAPIRE, R.E., SINGER, Y. (2000) Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, Journal of Machine Learning Research, 1, 113-141.

5. AMARI, S. (1967) Theory of Adaptive Pattern Classifiers. IEEE Transactions on Electronic Computers, 16, 299-307.

6. ANQUETIL, E. L., G. (1997) Perceptual model of handwriting drawing. Application to thehandwriting segmentation problem. Fourth International Conference on Document Analysis and Recognition, 1997.

7. ARTIERES, T. G., P. (2002) Stroke level HMMs for on-line handwriting recognition. Eighth International Workshop on Frontiers in Handwriting Recognition, 2002. .

8. ARTIERES, T. M., J.-M.; GALLINARI, P.; DORIZZI, B. (2000) Multi-modal segmental models for online handwriting recognition. 15th International Conference on Pattern Recognition.

9. ASUNCION, A. N., D.J. (2007) UCI Machine Learning Repository , (). [Irvine, CA: University of California, School of Information and Computer Science.

10. AUGUSTIN, E., BARET, O., PRICE, D. AND KNERR, S (1998) Legal Amount Recognition on French Bank Checks Using a Neural Network-Hidden Markov Model Hybrid. International Workshop on Frontiers in Handwriting Recognition.

11. AWAL, A. M., COUSSEAU, R., VIARD-GAUDIN, C. (2008) Convertisseur d'équations LATEX2Ink. Colloque International Francophone sur l'Ecrit et le Document. Rouen, France.

12. BAHL, L. R., BROWN, P.F., SOUZE, P.V.DE, AND MERCER, R.L. (1986) Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. ICASSP 86.

13. BAHL, L. R., BROWN, P.F., SOUZE, P.V.DE, AND MERCER, R.L. (1992) Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. ICASSP 86.

14. BAHLMANN, C. B., H. (2004) The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. Ieee Transactions on Pattern Analysis and Machine Intelligence, 26, 299-310.

15. BAHLMANN, C. H., B. BURKHARDT, H. (2002) Online handwriting recognition with support vector machines - a kernel approach. Eighth International Workshop on Frontiers in Handwriting Recognition, 2002.

16. BAKER, J. (1975) Stochastic Modeling as a Means of Automatic Speech Recognition. Carnegie Mellon University.

17. BAUM, L. E. P., T. , SOULES, G. AND WEISS N. (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist, 41, 164 -171.

18. BEIGI, H. M. S., NATHAN, K., SUBRAHMONIA, J. (1995) Online Uncostraint Handwriting Recognition Based on Probabilistic Techniques.

19. BEIGI, H. S. M. N., K. CLARY, G.J. SUBRAHMONIA, J. (1994) Size normalization in on-line unconstrained handwriting recognition. IEEE International Conference Image Processing, 1994. ICIP-94., . Austin, TX, USA.

20. BELLEGARDA, E. J. B., J.R. NAHAMOO, D. NATHAN, K.S. (1994) A fast statistical mixture algorithm for on-line handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16, 1227-1233.

21. BELLILI, A., GILLOUX, M., GALLINARI, P. (2000) An Hybrid MLP-SVM Handwritten Digit Recognizer.

22. BENGIO, Y., AND LECUN, Y. (1995a) LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. Neural Computing,, 7.

23. BENGIO, Y., AND LECUN, Y. (1995b) LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. Neural Computing Surveys, 7.

24. BENGIO, Y., AND LECUN, Y. (1994) Word Normalization for On-Line Handwritten Word Recognition. International Conference on Pattern Recognition, ICPR'94. Jurusalem, Israel.

25. BENGIO, Y., LECUN, Y., AND HENDERSON, D. (Ed.) (1993) Globally Trained Handwritten Word Recognizer Using Spatial Representation, Space Displacement Neural Networks and Hidden Markov Models, San Mateo CA.

26. BENGIO, Y., MORI, R.DE, FLAMMIA, G., AND KOMPE, R. (1991) Global Optimization of a Neural Network-Hidden Markov Model Hybrid. International Joint Conference on Neural Networks. Seattle.

27. BENTOUNSI, H., BATOUCHE, M. (2004) Incremental support vector machines for handwritten Arabic character recognition. International Conference on Information and Communication Technologies 2004.

28. BIADSY, F., EL-SANA, J., HABASH, N. (2006) Online Arabic Handwriting Recognition Using Hidden Markov Models. International Workshop on Frontiers in Handwriting Recognition, . La Baulle, France.

29. BIEM, A. (2001) Minimum classification error training for online handwriting recognition. Ieee Transactions on Pattern Analysis and Machine Intelligence, 28, 1041-1051.

30. BIEM, A. (2006) Minimum Classification Error Training for Online Handwriting Recognition IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, 1041-1051.

31. BIPPUS, R., MARGNER, V. (1999) Script recognition using inhomogeneous p2dhmm and hierarchical search space reduction. 5th

International Conference on Document Analysis and Recognition (ICDAR1999). Bangalore, India.

32. BISHOP, C. M. (1996) Neural Networks for Pattern Recognition, Oxford Univ Press, USA.

33. BODENHAUSEN, U., MANKE, S., WAIBEL, A. (1993) Connectionist Architectural Learning for High Performance Character and Speech Recognition. International Conference on Acoustics Speech and Signal Processing. Minneapolis, MN, USA.

34. BORTOLOZZI, F., BRITTO JR,. A., OLIVEIRA, L. S. AND MORITA, M., (Ed.) (2005) Recent Advances in Handwriting Recognition. .

35. BOULARD, H., BENGIO, S. (2003) Hidden Markov Model, London England, MIT Press.

36. BOURLARD, H., AND MORGAN, N. (Ed.) (1998) Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions.

37. BRAKENSIEK, A., WILLETT, D., RIGOLL, G. (2000) Unlimited vocabulary script recognition using character n-grams. 22. DAGM-Symposium. Kiel, Germany, Tagungsband Springer-Verlag.

38. BUNKE, H., ROTH, M., AND SCHUKAT-TALAMAZZINI, E.G. (1995) Off-line cursive Handwriting Recognition using Hidden Markov Models. Pattern Recognition, 28, 1399--1413.

39. BURGES, C. J. (1996) Simplified support vector decision rules. IN SAITTA, L. (Ed. 13th International Conference on Machine Learning. San Mateo, California., Morgan Kaufmann.

40. BURGES, C. J. C. (1998) A Tutorial on Support Vector Machines for Pattern Recognition. Kluwer Academic Publishers, Boston.

41. BURGES, C. J. C., BEN, J. I., DENKER, J. S., LECUN, Y. , NOHL, C. R. (1993) Off-line recognition of handwritten postal words using neural networks. International Journal of Pattern Recognition and Artificial Intelligence, 7, 689-704.

42. CAI, J., LIU, Z. Q. (1993) Off–line unconstrained handwritten word recognition. International Journal of Pattern Recognition and Artificial Intelligence, 14, 259–280.

43. CAILLAULT, E. (2005) Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne. Ecole Doctoral Sciences et Technologie de l'Information et des Materiaux (EDSTIM). Nantes, University of Nantes.

44. CAILLAULT, E., VIARD-GAUDIN, C. (2006) Using Segmentation Constraints in an Implicit Segmentation Scheme for On-line Word Recognition. 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006) La Baule, France.

45. CAMASTRA, F. (2007) A SVM-based cursive character recognizer. Pattern Recognition, 40, 3721-3727.

46. CHAKRAVARTHY, V. S., CHANDRASEKHAR, C. (2007) Online Handwritten Character Recognition Systems for Devanagari, Telugu and Kannada. Online Handwritten Character Recognition Consortium. Madras, India, Indian Institute of Technology.

47. CHANG, C. C., LIN, C. J. (2001) LIBSVM: a library for support vector machines.

48. CHAPELLE, O. (1998) Support Vector Machines for Image Classification. Stage de deuxieme annee Lyon, Ecole Normale Supeerieure de Lyon & Image Processing Reseach, AT&T, Redbank, NJ, USA.

49. CHAPELLE, O. H., P. VAPNIK, V.N. (1999) Support vector machines for histogram-based image classification. IEEE Transactions on Neural Networks, 10, 1055-1064.

50. CHEN, M. Y., KUNDU, A., AND ZHOU, J. (1994) Off-Line Handwritten Word Recognition Using Hidden Markov Model Type Stochastic Network. IEEE Trans on PAMI, 16, 481-496.

51. CHEN, M. Y., KUNDU, A., SRIHARI, AND S.N. (1995) Variable Duration Hidden Markov Mode and Morphological segmentation for Handwritten Word Recognition. IEEE Trans. Image Processing, 14, 1675-1687.

52. CHO, S.-B. (1995) On-line handwriting recognition with a neuro-fuzzy method. Fourth IEEE International Conference on Fuzzy Systems 1995 and The Second International Fuzzy Engineering Symposium.

53. CHO, W., KIM, J. H. (1994) Off-line recognition of cursive words with network of hidden markov models. 4th International Workshop on the Frontiers of Handwriting Recognition. Taipei, Taiwan.

54. CHOW, Y. L., DUNHAM, M. O., KIMBALL, O. A., KRASNER, M. A., KUBALA, G. F., MAKHOUL, J., ROUCOS, S., SCHWARTZ, R. M. (1987) BYBLOS: The BBN Continuous Speech Recognition System. IEEE International Conference on Acoustics, Speech, and Signal Processing, 1987.

55. CHU, F., WANG, L. (2005) Application of Support Vector Machines to Cancer Classification with Microarray Data. International Journal of Neural Systems, World Scientific Publishing Company, 15, 475-484.

56. COLLOBERT, R. (2001) SVMTorch: Support Vector Machines for Large-Scale Regression Problems. Journal of Machine Learning Research, 1, 143 - 160.

57. COVER, T. M., THOMAS, J.A. (1991) Elements of Information Theory, New York, USA, John Wiley and Sons.

58. DIMAURO, G., IMPEDOVO, S., PIRLO, G., SALZO, A. (Ed.) (1997) Automatic bankcheck processing: A new engineered system, World Scientific.

59. DOWNS, T., GATES, K.E. , MASTERS, A. (2001) Exact Simplification of Support Vector Solution. International Journal of Machine Learning Research, 2, 293-297.

60. DRUCKER, H. D. W. V., V.N. (1999) Support vector machines for spam categorization. IEEE Transactions on Neural Networks, 10, 1048-1054

61. DUDA, R. O., HART, P.E., AND STORK, D.G. (2001) Pattern Classification, John Wiley & son.

62. DZUBA, G., FILATOV, A., GERSHUNY, D., KILL, I. (1998) Handwritten word recognition - the approach proved by practice. 6th International Workshop on Frontiers in Handwriting Recognition. Taejon, Korea.

63. EDSON, J. R., BORTOLOZZIA, J. F., SABOURIN, R. (2005) A comparison of SVM and HMM classifiers in the off-line signature verification. Pattern Recognition Letters, 26, 1377-1385.

64. FAROUZ, C., GILLOUX, M., AND BERTILLE, J-M. (1998) Handwritten Word Recognition with Contextual Hidden Markov Models. International Workshop on Frontiers in Handwriting Recognition. Taejon.

65. FAVATA, J. T. (2001) Offline general handwritten word recognition using an approximate beam matching algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23, 1009-1021.

66. FRIESS, T., CRISTIANINI, N. AND CAMPBELL C. (1998) The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. 15th International Conference on Machine Learning. Morgan Kaufman Publishers.

67. FU, K. S. (1982) Syntactic Pattern Recognition and Applications., Englewood Cliffs, N.J., Prentice-Hall,.

68. GADER, P., MOHAMED, M., AND CHIANG, J.-H. (1994) Handwritten Word Recognition with Character and Inter-character Neural Networks. IEEE. Trans. System Man and Cybernetics, 27, 158-164.

69. GADER, P., WHALEN, M., GANZBERGER, M., HEPP, D. (1995) Handprinted word recognition on a NIST data set. Machine Vision and Applications, 8, 31– 41.

70. GADER, P. D., FORESTER, B., GANZBERGER, M., BILLIES, A., MITCHELL, B., WHALEN, M., YOUCUM, T. (1991) Recognition of handwritten digits using template and model matching. Pattern Recognition, 5, 421-431.

71. GADER, P. D., KELLER, J. M. (1996) Fuzzy Methods in Handwriting Recognition: An Overview. Biennial Conference of the

North American Fuzzy Information Processing Society, 1996. NAFIPS. 1996 137 - 141.

72. GANAPATHIRAJU, A. (2002) Support Vector Machine For Speech Recognition. Department of Electrical and Computer Engineering. Mississippi, Mississippi State University.

73. GANAPATHIRAJU, A. H., J.E.; PICONE, J. (2004) Applications of support vector machines to speech recognition. IEEE Transactions on Signal Processing, 52, 2348 - 2355.

74. GARCIA-SALICETTI, S., DORIZZI, B., GALLINARI, P., WIMMER, Z. (1996) Discriminative Training of a Neural Predictive System for on-line Word Recognition. Symposium on control, optimization and supervision. Lille.

75. GILLOUX, M., LEMARIÉ, B., AND LEROUX, M. (1995) A Hybrid Radial Basis Function/Hidden Markov Model Handwritten Word Recognition System. International Conference on Document Analysis and Recognition (ICDAR). Montreal.

76. GUILLEVIC, D. A. S., C.Y. (1995) Cursive Script Recognition Applied to the Processing of Bank Cheques. International Conference on Document Analysis and Recognition (ICDAR). Montreal.

77. GUYON, I., SCHENKEL, M., AND DENKER, J. (Ed.) (1996) Overview and Synthesis of On-line Cursive Handwriting Recognition Techniques, World Scientific Publishing Company.

78. GUYON, I., SCHOMAKER, L., PLAMONDON, R., LIBERMAN, M. & JANET, S. (1994) UNIPEN project of on-line data exchange and recognizer benchmarks. 12th International Conference on Pattern Recognition, ICPR'94. Jerusalem, Israel, IAPR-IEEE.

79. HA, J., OH, S., KIM, J. AND KWON, Y. (1993) Unconstrained handwritten word recognition with interconnected hidden Markov models. Third International Workshop on Frontiers in Handwriting Recognition, IAPR. Buffalo.

80. HASTIE, T. A. T., R. (1996) Classification by pairwise coupling. Technical report,. Stanford University and University of Toronto.

81. HSU, C. W. A. L., C.J (2002) A Comparison of Methods for Multiclass Support Vector Machines. IEEE TRANSACTIONS ON NEURAL NETWORKS, 13.

82. HU, J., BROWN, M.K. (1996) On-line handwriting recognition with constrained N-best decoding. Proceedings of the 13th International Conference on Pattern Recognition.

83. HUANG, W., NAKAMORIA, Y., WANG, S. Y. (2005) Forecasting stock market movement direction with support vector machine. Computers & Operations Research: Applications of Neural Networks, 32, 2513-2522.

84. HUANG, X. D., ARIKI, Y. AND JACK, M. (1990) Hidden Markov Models for Speech Recognition, Edinburgh University Press.

85. JAAKKOLA, T., DIEKHANS, M., HAUSSLER, D. (1999) Using the Fisher kernel method to detect remote protein homologies. IN AL, T. L. E. (Ed. 7th Int. Conference on Intelligent Systtem for Molecular Biology (ISMB-99).

86. JAIN, A. K., DUIN, R.P.W., AND MAO, J. (2000) Statistical Pattern Recognition: A Review. IEEE Trans. on PAMI, 22, 4-37.

87. JELINEK, F. (1976) Continuous speech recognition by statistical methods. Proceedings of the IEEE.

88. JOACHIMS, T. (1998) Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. ECML 1998.

89. JOACHIMS, T. (1999) Making large-Scale SVM Learning Practical. IN SCHÖLKOPF, B. B., C. AND SMOLA, A. (Ed.) Advances in Kernel Methods - Support Vector Learning. MIT-Press.

90. JUANG, B.-H. K., S. (1992) Discriminative learning for minimum error classification. IEEE Transactions on Signal Processing, 40, 3043 - 3054.

91. KARUSH, W. (1939) Minima of Functions of Several Variables with Inequalities as Side Constraints. Dept. of Mathematics. Chicago, Illinois, Univ. of Chicago.

92. KEERTHI, S. S., LIN, C.-J. (2003) Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation 15, 1667 - 1689.

93. KEERTHI, S. S., SHEVADE, S. K., BHATTACHARYYA, C AND MURTHY, K. R. K. (1999a) Improvement to Platt's SMO Algorithm for SVM Classifier Design. Singapore, National University of Singapore.

94. KEERTHI, S. S., SHEVADE, S. K., BHATTACHARYYA, C AND MURTHY, K. R. K. (1999b) A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. Intelligent Systems Lab, Indian Institute of Science, Bangalore.

95. KIM, G., GOVINDARAJU, V. (Ed.) (1997) Bankcheck recognition using cross validation between legal and courtesy amounts, World Scientific.

96. KIM, H. Y., KIM, J.H. (1998) Handwritten korean character recognition based on hierarchical randon graph modeling. 6th

International Workshop on Frontiers of Handwriting Recognition. Taegon-Korea.

97. KLEIN, D. (2000) Lagrange Multipliers without Permanent Scarring. University of Berkeley.

98. KOERICH, A. L., LEYDIER, Y., SABOURIN, R., SUEN, C. Y. (2002) A Hybrid Large Vocabulary Handwritten Word Recognition System using Neural Networks with Hidden Markov Models.

99. KOHONEN, T. (1988) Self-Organization and Associative Memory, Springer.

100. KOWALCZYK, A. (2001) A Gradient Based Training Method for A Support Vector Machine. IN LTD., T. R. D. M. P. (Ed. Patent Office, Australia, Canberra. G06F 15/18 ed. Australia.

101. KUHN, H. W. T., A. W. (1951) Nonlinear programming. 2nd Berkeley Symposium. Berkeley, University of California Press.

102. KUHN, M. (2006) The Karush-Kuhn-Tucker Theorem. Mannheim, Germany, CDSEM Uni Mannheim.

103. KULLBACK, S. (1997) Information Theory and Statistics, Mineola, New York, Dover Publications.

104. KUNDU, A., BAHL, L. (1988) Recognition of handwritten script: a hidden Markov model based approach. International Conference on Acoustics, Speech and Signal Processing. New York.

105. KWOK, J. (1999) Moderating the Outputs of Support Vector Machine Classifiers. IEEE Trans. on Neural Networks, 10.

106. LALLICAN, P. M. (1999) Reconnaissance de l'Ecriture Manuscrite Hors-ligne : Utilisation de la Chronologie Restaurée du Tracé.

IRESTE, Ecole Doctorale Sciences pour l'Ingénieur de Nantes. Universite de Nantes.

107. LECUN, Y., BOTOU, L., JACKEL, L., DRUCKER, H., CORTES, C., DENKER, J., GUYON, I., MÜLLER, U., SÄCKINGER, E., SIMARD, P. AND VAPNIK, V. (1995) Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition. Neural Networks, 261--276.

108. LECUN, Y., BOTTOU, L., BENGIO, Y., HAFFNER, P. (1998a) Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, 2278-2324. .

109. LECUN, Y., BOTTOU, L., ORR, G.B. AND MULLER, K.R. (Ed.) (1998b) Efficient Backprop, Springer Lecture Notes in Computer Sciences

110. LEE, K. F. (1988) Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system. Department of Computer Science. Pittsburg, Pennsylvania, Carnegie-Mellon University.

111. LEEDHAM, C. G. (1994) Historical perspectives of Handwriting Recognition Systems. IEE Journal, UK.

112. LETHELIER, L., M., GILLOUX, M. (1995) An automatic reading system for handwritten numeral amounts on french checks. 3rd International Conference on Document Analysis and Recognition. Montreal-Canada.

113. LEVINSON, S., RABINER, L. AND SONDHI, M. (1983) An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. Bell System Technical Journal, 64, 1035 - 1074.

114. LI, S. Z., QINGDONG, GU, F. L., SCHOLKOPF, B., CHENG, Y., ZHANG, H. (2001) Kernel machine based learning for multi-view face detection andpose estimation. Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Vancouver, BC, Canada.

115. LIN, H.-T., LIN, C.-J. (2003) A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Taipei, Department of Computer Science, National Taiwan University.

116. LIU, C.-L. J., S. NAKAGAWA, M. (2004) Online recognition of Chinese characters: the state-of-the-art. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26, 198 - 213.

117. LUNG, J. Z. (2004) SVM Multi-classifier and web document Classification. Third International Conference on Machine Learning and Cybernetics. Shanghai, Chine.

118. MADHVANATH, S., AND GOVINDARAJU, V. (2000) The Role of Holistic Paradigms in Handwritten Word Recognition. IEEE Transactions on PAMI, 23, 149-164.

119. MADHVANATH, S., KLEINBERG, E., GOVINDARAJU, V. (1999) Holistic verification of handwritten phrases. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21, 1344-1356.

120. MALAVIYA, A. P., L. THEISSINGER, M. (1994) FOHDEL-a new fuzzy language for online handwriting recognition. Third IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence. Orlando, FL, USA.

121. MARMELSTEIN, P., AND EDEN, M. (1964) A System for Automatic Recognition of Handwritten Word. FJCC (AFIPS).

122. MARTI, U., BUNKE, H. (2000) Handwritten sentence recognition. 15 th International Conference on Pattern Recognition. Barcelona, Spain.

123. MATAN, O., BURGES, C.J.C., LECUN, Y. AND DENKER, J.S. (1992a) Multi-digit Recognition Using a Space Displacement Neural Network. Neural Information Processing Systems.

124. MATAN, O., BURGES, J. C., LECUN, Y. , DENKER, J. S. (Ed.) (1992b) Multi-digit recognition using a space displacement neural network, Morgan Kaufmann.

125. MICO, L., ONCINA, J. (1999) Comparison of fast nearest neighbour classifier for handwritten character recogniton. Pattern Recognition Letters, 19, 351-356.

126. MOHAMED, M. A., GADER, P. (2000) Generalized hidden markov models - part ii: Application to handwritten word recognition. IEEE Transactions on Fuzzy Systems, 8, 82–94.

127. MOHAMED, M. G., P. (1996) Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18.

128. NAG, R., WONG, K., FALLSIDE, F. (1986) Script recognition using hidden Markov models. International Conference on Acoustics, Speech and Signal Processing. Tokyo.

129. NAKAGAWA, M. (1990) Non-Keyboard Input of Japanese Text-On-Line Recognition of Handwritten Characters as the Most Hopeful Approach. Journal of Information Processing, 13, 15 - 34.

130. NEWMAN, D. J. H., S. & BLAKE, C.L. & MERZ, C.J. (1998) UCI Repository of machine learning databases

131. NEY, H., MERGEL, D., NOLL, A. , PAESELER, A. (1987) A data-driven organization of the dynamic programming beam search for continuous speech recognition. Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07). Dallas, Texas.

132. OSUNA, E., FREUND, R. AND GIROSI, F. (1997) Training support vector machines: An application to face detection. Computer Vision and Pattern Recognition (CVPR'97). New York.

133. OSUNA, E. E., FREUND, R. AND GIROSI, F. (1996) Support Vector Machines : Training and Applications. MIT Report.

134. OUDOT, L. (2003) Fusion of information and adaptation for the recognition of dynamic handwritten texts (French). Informatique. Paris, Universit´e Pierre & Marie Curie.

135. PARIZEAU, M., AND PLAMONDON, R. (1995) A Fuzzy Syntactical Approach to Allograph Modelling for Cursive Script Recognition. IEEE Transactions on PAMI, 17, 702-712.

136. PLAMONDON, R., AND SRIHARI, S.N. (2000) On-Line and Off-line Handwriting Recognition: A Comprehensive Survey. IEEE Transactions on PAMI, 22, 63-84.

137. PLAMONDON, R., LORETTE, G. (1989) Automatic signature verification and writer identification - The state of the art. Pattern Recognition, 22, 107-131.

138. PLATT, J. (1998a) Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Res. Tech. Rep. , 98-114.

139. PLATT, J. (Ed.) (1999a) Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, Cambridge, Massachusettes, USA, MIT Press.

140. PLATT, J. C. (Ed.) (1998b) Fast training of support vector machines using sequential minimal optimization, Cambridge, MA, MIT Press.

141. PLATT, J. C. (Ed.) (1999b) Using Analytic QP and Sparseness to Speed Training of Support Vector Machines.

142. POISSON, E., VIARD-GAUDIN, C., LALLICAN, P.M. (2002) Multi-modular architecture based on convolutional neural networks for online handwritten character recognition. 9th International Conference on Neural Information Processing, ICONIP'02 - IEEE Neural Network Society,. Singapore.

143. PROCTER, S., ILLINGWORTH, J. (2000) Cursive Handwriting Recognition using Hidden Markov Models and a Lexicon-driven Level Building Algorithm. IEEE Proc. On Vision, Image and Signal Processing, 147.

144. RABINER, L., JUANG, B.-H. (1993) Fundamentals of Speech Recognition, Englewood Cliffs, New Jersey, Prentice-Hall, Inc.

145. RABINER, L. A. J., B. (1986a) An introduction to hidden Markov models. IEEE ASSP Magazine, 257 - 285.

146. RABINER, L. R., JUANG, B.H., LEVINSON, S.E. , SONDHI, M.M. (1985) Recognition Of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities. AT&T Technical Journal, 21-31.

147. RABINER, L. R. A. J., B.H. . (1986b) An introduction to hidden Markov models. IEEE ASSP Magazine, 4-16.

148. RATZLAFF, E. H. (2003) Methods, reports and survey for the comparison of diverse isolated character recognition results on the UNIPEN database. Seventh International Conference on Document Analysis and Recognition, 2003. . Edinburgh, Scotland.

149. RIGOLL, G. (1998) Hybrid Speech Recognition Systems: A Real Alternative to Traditional Approaches? Survey Lecture, Proc. International Workshop Speech and Computer (SPECOM'98). St. Petersburg, Russia.

150. RIGOLL, G. K., A. RATTLAND, J. NEUKIRCHEN, C. (1996) A Comparison Between Continuous and Discrete Density Hidden Markov Models for Cursive Handwriting Recognition. 13th International Conference on Pattern Recognition, 1996. Vienna, Austria.

151. RIIS, S. K. (1998) Hidden Markov Models and Neural Networks for Speech Recognition.

152. RUMELHART, D. E., HINTON, G.E. , R.J. WILLIAMS (Ed.) (1986) Learning Internal Representation by Error Propagation, Cambridge, MA, MIT Press.

153. SANGUANSAT, P., ASDORNWISED, W., JITAPUNKUL, S. (2004) Online Thai handwritten character recognition using hidden Markov models and support vector machines. International Symposium on Communications and Information Technologies, ISCIT 2004. Japan.

154. SAON, G. (1999) Cursive word recognition using a random field based hidden markov model. International Journal on Document Analysis and Recognition, 1, 199-208.

155. SCAGLIOLA, C., NICCHIOTTI, G. (2000) Enhancing cursive word recognition performance by the integration of all the available information. 7th International Workshop on Frontiers in Handwriting Recognition. Amsterdam, Netherlands.

156. SCHENKEL, M., GUYON, I, HENDERSON, D. (1995) Online Cursive Script Recognition using Time Delay Neural Networks and Hidden Markov Models. Machine Vision and Applications, 215-223.

157. SCHENKEL, M., WEISSMAN, H., GUYON, I. , NOHL, C. , AND HENDERSON, D. (Ed.) (1993) Recognition-based segmentation of on-line hand-printed words, Denver.

158. SCHOLKOPF, B., BURGES, C. AND SMOLA, A. (Ed.) (1999) Advances in Kernel Methods: Support Vector Learning. , Cambridge, MA, MIT Press.

159. SCHURMANN, J. (1996) Pattern Classification - A unified view of statistical and neural approaches, Wiley interscience.

160. SCHWAIGHOFER, A. A. T., V. (2001) The Bayesian committee support vector machine. IN DORFFNER, G., BISCHOF, H. AND HORNIK, K. (Ed. Artificial Neural Networks - ICANN 2001. Springer Verlag.

161. SHEVADE, S. K., KEERTHI, S. S., BHATTACHARYYA, C. AND MURTHY, K. R. K. (2000) Improvements to the SMO Algorithm for SVM Regression, IEEE TRANSACTIONS ON NEURAL NETWORKS, 11.

162. SHRIDHAR, M., BADRELDIN, A. (1986) Recognition of isolated and simply connected handwritten numerals. Pattern Recognition, 19, 1-12.

163. SIDENBLADH, H. (2004) Detecting Human Motion with Support Vector Machines. 17th IAPR International Conference on Pattern Recognition. Cambridge, UK.

164. SOONG, F. K. H., E.-F. (1991) A tree-trellis based fast search for finding the N-bestsentence hypotheses in continuous speech recognition. International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991.

165. STEINHERZ, T., RIVLIN, E., AND INTRATOR, N. (1999) Off-Line Cursive Script Word Recognition - A Survey. Int'l Journal of Document Analysis and Recognition.

166. TAPPERT, C. C., SUEN, C. Y., WAKAHARA, T. (1994) The state of the art in on-line handwriting recognition. IEEE Transactions on Pattern Analysis and Artificial Intelligence, 12, 787 - 808.

167. TAPPERT, C. C., SUEN, C.Y., AND WAKAHARA, T. (1990) The State of the Art in On-Line Handwriting Recognition. IEEE Trans. On PAMI, 12.

168. TAPPERT., C. C., SUEN, C. Y., WAKAHARA, T (1988) Online Handwriting Recognition - A Survey. 9th International Conference on Pattern Recognition, 1988. Rome, Italy.

169. TAY, Y. H. (2002) Offline Handwriting Recognition using Artificial Neural Network and Hidden Markov Model. Electrical Engineering. Johor Bahru, Universiti Teknologi Malaysia and Ecole Polytechnic University of Nantes.

170. TAY, Y. H., KHALID, M., YUSOF, R., VIARD-GAUDIN, C. (2003) Offline Cursive Handwriting Recognition System based on Hybrid Markov Model and Neural Network. IEEE Int'l Symp on

Computational Intelligence in Robotics and Automation (CIRA-2003) Kobe, Japan.

171. TRESP, V. (2000) A Bayesian Committee Machine. Neural Computation, 12, 2719-2741.

172. VAN GESTEL, T. S., J.A.K. BAESTAENS, D.-E. LAMBRECHTS, A. LANCKRIET, G. VANDAELE, B. DE MOOR, B. VANDEWALLE, J. (2001) Financial time series prediction using least squares support vectormachines within the evidence framework. IEEE Transactions on Neural Networks, 12, 809-821.

173. VAPNIK, V. (1995) The Nature of Statistical Learning Theory, New York, Springer-Verlag.

174. VAPNIK, V. (1998) Statistical Learning Theory, New York, Wiley.

175. VAPNIK, V. (1999) An Overview of Statistical Learning Theory. IEEE Transactions On Neural Networks, 10.

176. VIARD-GAUDIN, C., LALLICAN, P.M., KNERR, S., AND BINTER, P. (1999) The IRESTE On/Off (IRONOFF) Dual Handwriting Database. International Conference on Document Analysis and Recognition. Bangalore.

177. WAKAHARA, T., MURASE, H., ODAKA, K. (1992) On-Line Handwriting Recognition. Proc. IEEE-ICPR, 80, 1181-1194.

178. WAN, V. R., S. (2005) Speaker verification using sequence discriminant support vector machines. IEEE Transactions on Speech and Audio Processing, 13, 203- 210.

179. WATANABE, S. (1985) Pattern Recognition: Human and Mechanical. NewYork, Wiley.

180. WATKINS, C. (Ed.) (2000) Dynamic alignment kernels. , MIT Press.

181. WEIBEL, A., HANAZAWA, T., HINTON, G., SHIKANO, K., AND LANG, K. (1989) Phoneme Recognition Using TimeDelay Neural Networks. IEEE Transactions on Acoustic, Speech and Signal Processing, 37, 328-339.

182. WESTON, J., WATKINS, C. (1998) Multiclass Support Vector Machines. Technical Report CSD-TR-98-04. Egham, Surrey, England, Department of Computer Science, Royal Holloway,.

183. XIE, S. L., SUK, M. (1988) On machine recognition of hand-printed chinese character by feature relaxation. Pattern Recognition, 21, 1-7.

184. XU, F., LIU, X., FUJIMURA, K. (2005) Pedestrian detection and tracking with night vision. IEEE Transactions on Intelligent Transportation Systems, 6, 63- 71.

185. ZHANG, G. P. (2000) Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 30, 451-462.

186. ZHANG, M. F., YAN, H., FABRI, M. A. (1999) Handwritten digit recognition by adaptativesubspace self organizing map. . IEEE Trans. on Neural Networks, 10, 939-945.

187. ZHANG, M. G. (2005) Short-term Load Forecasting Based On Support Vector Machines Regression Fourth International Conference on Machine Learning and Cybernetics. Guangzhou.

188. ZHANG, M. G., YAN, W. W., YUAN, Z. T. (2004) Study of Nonlinear System identification Based on Support Vector Machine. Third International Conference on Machine Learning and Cybernetics. Shanghai, China.

APPENDIX A

PUBLISHED PAPERS

1. **Ahmad, A. R.**, Wahap, A. R. Khalid, M., Yusof, R. (2002). A Neural Network Based Bank Cheque Recognition system for Malaysian Cheques International Conference on AI in Engineering and Technology, ICAIET 2002, Kota Kinabalu, Sabah, Malaysia.

2. **Ahmad, A. R**., Wahap, A. R. Khalid, M., Yusof, R. (2002). Machine Learning Using Support Vector Machine. Malaysian Science and Technology Conference, MSTC 2002. Johor Bahru, Malaysia.

3. **Ahmad, A. R.**, Khalid, M., Yusof, R. (2002). Support Vector Machine and Kernel methods for Handwriting Recognition. Student Conference in Research and Development SCORED 2002. Shah Alam, Malaysia.

4. **Ahmad, A. R.**, Khalid, M., Yusof, R., Viard-Gaudin, C. (2003). Comparative Study of SVM Tools for Data Classification. First Regional Malaysia-France Workshop on Image Processing In Vision Systems and Multimedia Communication, IMAGE2003. Sarawak, Malaysia.

5. **Ahmad, A. R.**, Khalid, M., Yusof, R., Viard-Gaudin, C. (2004). Online Handwriting Recognition using Support Vector Machine and Hidden Markov Model. International Conference on AI in Engineering and Technology, ICAIET 2004. Sabah, Malaysia.

6. **Ahmad, A. R**., Khalid, M., Viard-Gaudin, C. (2004). "Comparison of Support Vector Machine and Neural Network in Character Level Discriminant Training

for Online Word Recognition." UNITEN Students Conference on Research and Development 2004, SCORED2004.

7. **Ahmad, A. R**., Khalid, M., Viard-Gaudin, C., Poisson, E. (2004). Online Handwriting Recognition Using Support Vector Machine. IEEE Region 10 Conference TENCON 2004. . Chiangmai Thailand. 1: 311- 314.

8. Cailllaut, E., Viard-Gaudin, C. **Ahmad, A. R.** (2005). MS-TDNN with Global Discriminant Trainings. Eighth International Conference on Document Analysis and Recognition (ICDAR'05), Seoul, Korea.

9. Syed Ahmad, S. M., **Ahmad, A. R.**, Shakil, A., Md. Anwar, R., Begum, S. (2007). Hybrid Online and Offline Malaysian Signature and Malay Handwriting Data Collection. Conference on IT Research & Applications, CITRA 2007. Subang, Malaysia.

10. Syed Ahmad, S. M., **Ahmad, A. R.**, Shakil, A., Mustafa Agil, M. B., Md. Anwar, R., Begum, S. (2008). SIGMA- A Malaysian Signature's Database. IEEE ACS International Conference on Computer Systems and Applications (AICCSA2008). Doha, Qatar.

APPENDIX B

LAGRANGE MULTIPLIERS METHOD AND THE KARUSH-KUHN-TUCKER THEOREM

The Lagrange multipliers method (named after Joseph Louis Lagrange, a French Italian Mathematician) is the basic tool in nonlinear constrained optimization. It is used to find the extrema of a function of several variables subject to one or more constraints. A set of conditions, the Karush–Kuhn–Tucker (KKT) conditions are necessary for a solution in the optimization to be optimal, provided some regularity conditions are satisfied. KKT conditions were first published in the Master's thesis of William Karush (Karush, 1939), but they were renowned only after a seminal conference paper by Kuhn and Tucker (Kuhn, 1951).

The Lagrange multipliers method is able to determine where on a particular set of points a particular function is the optimum. The stationary points of the constrained function are computed. By Fermat's theorem, extrema occur either at these points, or on the boundary, or at points where the function is not differentiable. Finding stationary points of a constrained function in n variables with k constraints is reduced to finding stationary points of an unconstrained function in n+k variables.

An unknown scalar variable (called the Lagrange multiplier) is introduced for each constraint, and a new function is defined (called the Lagrangian) in terms of the original function, the constraints, and the Lagrange multipliers.

1. Problem formulation and the Lagrange function

Consider the following maximization problem

$$\max_{x \in R^n} \quad f(x)$$

$$\text{such that } g_j(x) \geq 0 \quad j = 1,...,m$$

$$h_i(x) = 0 \quad i = 1,...,n$$

with $f: \Re^N \to \Re$, $g_j: \Re^N \to \Re^p$, $h_i: \Re^N \to \Re^M$ being continuously differentiable functions.

2. Saddle points of the Lagrangian and Karush-Kuhn-Tucker points

Define the Lagrange function of the problem as

$$L(x, \lambda, \mu) = f(x) + \sum_{j=1}^{m} \lambda_j g_j(x) + \sum_{i=1}^{n} \mu_i h_i(x)$$

Define a saddle point of the Lagrangian as a tuple

$$(\tilde{x}, \tilde{\lambda}, \tilde{\mu}) \text{ such that } L(\tilde{x}, \tilde{\lambda}, \tilde{\mu}) = \min_{\mu, \lambda \geq 0} \max_{x} L(x, \lambda, \mu)$$

We know that

$$\min_{\mu, \lambda \geq 0} L(x, \lambda, \mu) \leq \min_{\mu, \lambda \geq 0} \max_{x} L(x, \lambda, \mu) \leq \max_{\mu, \lambda \geq 0} L(x, \lambda, \mu)$$

i.e: $(\tilde{x}, \tilde{\lambda}, \tilde{\mu})$ is a critical point of $L(x, \lambda, \mu)$, but neither a minimum nor a maximum. As a next step we want to establish the connection between a saddle point and the solution to the maximization problem.

Consider the Lagrangian

$$L(x, \lambda, \mu) = f(x) + \sum_{j=1}^{m} \lambda_j g_j(x) + \sum_{i=1}^{n} \mu_i h_i(x)$$

and the first order condition (FOC) with respect to x, which characterize the critical points of $L(x, \lambda, \mu)$ and are necessary for a maximum

$$\nabla_x f(\tilde{x}) + \sum_{j=1}^{m} \lambda_j \nabla_x g_j(\tilde{x}) + \sum_{i=1}^{n} \mu_i \nabla_x h_i(\tilde{x}) = 0$$

Furthermore, consider the FOC of $L(\tilde{x},\lambda,\mu)$ with respect to (λ,μ), which are necessary for a minimum of $L(\tilde{x},\lambda,\mu)$

$$L(\tilde{x},\lambda,\mu) = f(\tilde{x}) + \sum_{j=1}^{m} \lambda_j g_j(\tilde{x}) + \sum_{i=1}^{n} \mu_i h_i(\tilde{x})$$

Define

$$d(\lambda,\mu) = L(\tilde{x},\lambda,\mu)$$

The function $d(\lambda,\mu)$ is also called the dual function of the problem. Notice that $d(\lambda,\mu)$ is and affine function independent of the functional form of f(x), gj(x), hi(x). Since it is a linear programming problem, the minimum of the function is either $f(\tilde{x})$ or it does not exist.

$$\min_{\mu,\lambda \geq 0} d(\lambda,\mu) = \begin{cases} f(\tilde{x}) & if \quad g_j(\tilde{x}) \geq 0 \quad \forall j \text{ and } h_i(\tilde{x}) = 0 \ \forall i \\ -\infty & else \end{cases}$$

From this result, we can conclude that every saddle point must be a solution to the original maximization problem. To see why, consider two arguments:

(a) A saddle point exists if and only if $\tilde{x}$ is feasible for the maximization problem, i.e: $g_j(\tilde{x}) \geq 0 \ \ \forall j \ \wedge \ h_i(\tilde{x}) = 0 \ \forall i$

(Existence saddle point $\Rightarrow$ feasibility of $\tilde{x}$)

(b) The Lagrange function with $(\tilde{\lambda},\tilde{\mu})$ overestimates the objective function on the interior of the feasible set. To see this, consider the following equivalent problem

$$\max_x \ f(x) + \sum_{j=1}^{m} I(g_j(x) \geq 0) + \sum_{i=1}^{n} I(h_i(x) = 0)$$

with

$$I(g_j(x)\geq 0) = \begin{cases} 0 & if \quad g_j(\tilde{x}) \geq 0 \quad \forall j \\ -\infty & else \end{cases}$$

$$I(h_i(x)=0) = \begin{cases} 0 & if \quad h_i(\tilde{x}) = 0 \quad \forall i \\ -\infty & else \end{cases}$$

This means we penalize the function for violations of the constraints. This problem is equivalent to the first, if we assume that a solution exists. In a next step, we replace the"hard" penalty function by"weak" linear penalty functions.

$$\max_x \; f(x) + \sum_{j=1}^{m} \tilde{\lambda}_j g_j(x) + \sum_{i=1}^{n} \tilde{\mu}_i h_i(x)$$

with

$$\tilde{\lambda}_j \geq 0, \quad \tilde{\mu}_j \geq 0 \quad \forall i, j$$

For feasible values, i.e

$$g_j(x) \geq 0 \text{ and } h_i(x) = 0$$

We clearly overestimate the true objective function. We, therefore, know that there are no other (feasible) choices $\hat{x}$ with a higher value of the objective function than the saddle point of the problem with value $f(\hat{x})$.

APPENDIX C

Verbose output of recognition and segmentation

```
Word dbdFileName    : C:\MyApps\dbd\hi.dbdnew.dbd
modelFileName       :
..\model\EnglishWord_train_unp.dbdnew.dbdchar.dbd.fe.mod
pspFileName         : C:\MyApps\dbd\hi.dbdnew.dbd_psp.txt
maxSlices chosen    : 5
Total examples : 1
wordLexiconFileName : ..\lex\Englishwordlexicon.lex
charLexiconFilename : ..\lex\characterlexicon.lex

Ex[1]: C:\myapps\unp\hi.unp_pre.unp ->hi

Stroke (0) --> Total points 129

Slice [0]: 40 points from stroke no. 0, point 0 to point 39
Slice [1]: 24 points from stroke no. 0, point 40 to point 63
Slice [2]: 14 points from stroke no. 0, point 64 to point 77
Slice [3]: 14 points from stroke no. 0, point 78 to point 91
Slice [4]: 16 points from stroke no. 0, point 92 to point 107
Slice [5]: 21 points from stroke no. 0, point 108 to point 128

Total slices = 6, Total hypothesis = 20
==========
Hypothesis [0]- num pts = 40, from slice 0, to 0
Hypothesis [1]- num pts = 24, from slice 1, to 1
Hypothesis [2]- num pts = 64, from slice 0, to 1
Hypothesis [3]- num pts = 14, from slice 2, to 2
Hypothesis [4]- num pts = 38, from slice 1, to 2
Hypothesis [5]- num pts = 78, from slice 0, to 2
Hypothesis [6]- num pts = 14, from slice 3, to 3
Hypothesis [7]- num pts = 28, from slice 2, to 3
Hypothesis [8]- num pts = 52, from slice 1, to 3
Hypothesis [9]- num pts = 92, from slice 0, to 3
Hypothesis [10]- num pts = 16, from slice 4, to 4
Hypothesis [11]- num pts = 30, from slice 3, to 4
Hypothesis [12]- num pts = 44, from slice 2, to 4
Hypothesis [13]- num pts = 68, from slice 1, to 4
Hypothesis [14]- num pts = 108, from slice 0, to 4
Hypothesis [15]- num pts = 21, from slice 5, to 5
Hypothesis [16]- num pts = 37, from slice 4, to 5
Hypothesis [17]- num pts = 51, from slice 3, to 5
Hypothesis [18]- num pts = 65, from slice 2, to 5
Hypothesis [19]- num pts = 89, from slice 1, to 5
```

```
The probability matrix (20 hypothesis x 50 classes in SVM model).
Classes : 3 32 47 50 41 4 28 43 29 36 39 52 5 45 30 42 8 49 34 9 31
12 48 1 51 13 14 15 35 16 46 17 38 18 53 19 20 40 24 25 26 27 2 6 7
10 11 21 22 23.
(Match class number against character using character lexicon file
characterlexicon.lex)

Hypothesis 0:
-2.84 -1.31 -1.60 -2.25 -1.72 -3.20 -2.60 -2.60 -2.75 -0.95 -2.68 -
2.79 -2.85 -0.97 -2.12 -1.36 -2.90 -1.51 -2.93 -2.78 -2.66 -2.88 -
0.57 -0.70 -2.62 -2.95 -2.52 -2.28 -2.85 -2.70 -1.27 -2.77 -2.81 -
2.87 -2.79 -3.02 -2.69 -2.57 -2.35 -2.83 -2.92 -2.85 -2.67 -2.81 -
2.77 -2.78 -2.87 -2.86 -2.35 -2.30

Hypothesis 1:

 -3.30 -1.92 -0.98 -3.45 -2.78 -2.89 -2.74 -2.32 -2.90 -1.22 -0.13 -
3.03 -3.18 -2.61 -1.66 -2.86 -3.06 -3.42 -3.08 -3.14 -3.09 -3.30 -
3.34 -2.75 -2.63 -1.91 -3.27 -3.34 -2.89 -3.24 -2.27 -3.26 -3.02 -
3.02 -2.92 -3.28 -2.76 -3.45 -3.44 -3.01 -2.84 -3.13 -3.31 -3.25 -
3.08 -2.26 -2.84 -3.31 -2.92 -3.35

Hypothesis 2:

 -3.50 -0.11 -1.34 -3.52 -3.00 -3.45 -2.83 -3.19 -3.14 -0.99 -1.37 -
2.99 -3.46 -2.60 -2.34 -2.50 -3.54 -3.38 -3.62 -3.31 -2.94 -3.52 -
3.03 -3.14 -2.62 -3.32 -2.95 -3.40 -3.38 -3.21 -2.72 -3.36 -3.44 -
3.05 -3.40 -3.61 -3.14 -3.43 -3.56 -2.99 -3.02 -3.49 -3.38 -3.58 -
3.43 -3.39 -3.51 -3.39 -3.02 -3.44

Hypothesis 3:

 -3.70 -2.03 -1.33 -3.27 -2.13 -3.88 -3.33 -3.32 -3.56 -1.32 -3.37 -
3.62 -3.70 -0.08 -1.99 -2.26 -3.71 -2.25 -3.75 -3.65 -3.43 -3.75 -
2.46 -1.63 -3.39 -3.41 -3.27 -3.27 -3.70 -3.54 -2.54 -3.55 -3.65 -
3.63 -3.57 -3.69 -3.14 -3.33 -3.32 -3.61 -3.74 -3.54 -3.53 -3.61 -
3.53 -3.59 -3.63 -3.55 -3.43 -2.97

Hypothesis 4:

 -4.08 -3.17 -0.03 -3.95 -3.68 -3.49 -2.49 -2.51 -2.95 -2.13 -2.36 -
4.08 -4.05 -2.73 -2.88 -2.84 -2.70 -3.86 -3.58 -3.41 -3.29 -4.17 -
3.94 -3.67 -3.54 -1.66 -4.14 -3.90 -3.35 -3.70 -3.14 -4.04 -4.00 -
3.81 -3.20 -4.13 -3.65 -4.18 -3.81 -3.99 -3.79 -3.45 -4.06 -4.04 -
3.93 -3.64 -3.18 -4.06 -3.56 -3.83

Hypothesis 5:

 -2.53 -0.61 -0.68 -2.35 -1.98 -2.92 -1.81 -2.17 -1.19 -1.12 -1.32 -
2.09 -2.57 -1.02 -2.07 -1.62 -2.53 -2.24 -2.44 -2.47 -2.16 -2.52 -
2.13 -2.49 -2.06 -2.63 -2.38 -2.47 -2.01 -2.35 -1.74 -2.60 -2.42 -
2.28 -2.39 -2.66 -1.67 -2.53 -2.46 -2.44 -2.23 -2.51 -2.19 -2.47 -
2.38 -2.31 -2.46 -2.52 -2.57 -2.62

Hypothesis 6:

 -3.81 -2.62 -1.08 -3.66 -2.93 -3.52 -3.38 -3.43 -3.15 -0.11 -1.74 -
3.89 -3.75 -1.98 -2.33 -3.69 -3.72 -3.47 -3.82 -3.65 -3.48 -3.84 -
3.58 -2.90 -2.02 -1.09 -3.84 -3.62 -2.78 -3.99 -3.37 -3.85 -3.38 -
3.67 -3.37 -3.88 -3.57 -3.71 -3.54 -3.70 -3.79 -3.46 -3.82 -3.92 -
```

```
3.91 -3.18 -3.64 -3.87 -3.65 -3.84

Hypothesis 7:

 -4.18 -3.02 -3.17 -4.12 -1.82 -4.39 -3.88 -4.05 -3.96 -1.21 -3.44 -
4.14 -4.38 -0.04 -3.19 -3.96 -4.49 -3.26 -4.50 -4.35 -4.11 -4.54 -
3.69 -3.73 -3.94 -3.67 -3.95 -4.24 -4.03 -4.11 -3.30 -4.37 -4.31 -
4.22 -4.27 -3.69 -3.81 -4.01 -4.35 -4.54 -4.26 -4.24 -4.06 -4.58 -
4.21 -4.29 -4.49 -4.50 -4.25 -3.81

Hypothesis 8:

 -3.77 -2.95 -2.53 -3.86 -2.54 -3.38 -2.41 -2.88 -2.24 -3.06 -3.27 -
3.79 -3.63 -2.95 -2.74 -3.15 -2.16 -3.41 -3.67 -3.43 -3.29 -3.50 -
3.08 -3.36 -2.24 -1.77 -3.36 -3.54 -0.04 -3.35 -3.38 -3.60 -1.92 -
3.05 -3.55 -3.63 -3.24 -3.45 -3.76 -2.67 -3.64 -3.30 -3.58 -3.56 -
3.68 -3.13 -3.35 -3.73 -2.97 -3.53

Hypothesis 9:

 -2.88 -1.70 -1.78 -2.52 -2.05 -3.36 -1.52 -2.48 -1.56 -1.89 -2.28 -
2.45 -2.89 -1.93 -2.02 -2.01 -2.87 -2.65 -2.74 -2.77 -2.63 -2.81 -
2.30 -2.88 -2.32 -2.87 -2.67 -2.73 -0.13 -2.80 -1.68 -2.93 -2.04 -
2.44 -2.65 -2.96 -2.38 -2.56 -2.68 -2.79 -2.61 -2.64 -2.26 -2.80 -
2.75 -2.58 -2.74 -2.79 -2.75 -2.90

Hypothesis 10:

 -3.14 -1.68 -1.41 -2.57 -1.82 -3.64 -2.77 -2.76 -2.99 -0.65 -2.97 -
3.12 -3.17 -0.31 -2.29 -1.67 -3.21 -1.49 -3.28 -3.07 -2.87 -3.19 -
1.25 -1.88 -2.87 -3.10 -2.68 -2.63 -3.23 -3.11 -1.40 -3.01 -3.13 -
3.17 -3.08 -3.31 -2.80 -2.94 -2.69 -3.10 -3.23 -3.17 -2.97 -3.05 -
3.00 -3.01 -3.12 -3.17 -2.80 -2.67

Hypothesis 11:

 -4.16 -3.53 -3.20 -3.07 -2.45 -4.36 -3.62 -3.99 -4.10 -1.10 -3.62 -
4.23 -4.16 -2.82 -3.28 -3.24 -4.06 -0.36 -4.31 -4.15 -4.16 -4.18 -
0.34 -2.72 -4.05 -3.65 -3.84 -3.80 -3.78 -3.61 -3.92 -4.17 -4.22 -
4.23 -4.11 -4.28 -3.99 -3.60 -2.60 -4.22 -4.29 -3.86 -4.30 -4.27 -
4.35 -4.30 -4.31 -4.30 -2.59 -2.84

Hypothesis 12:

 -3.53 -1.86 -1.98 -2.94 -1.68 -3.69 -3.13 -3.20 -3.00 -0.70 -2.85 -
3.45 -3.61 -0.26 -2.32 -2.37 -3.56 -0.99 -3.50 -3.68 -3.58 -3.77 -
1.30 -2.30 -3.38 -3.33 -3.32 -1.83 -3.12 -3.54 -2.75 -3.62 -3.44 -
3.54 -3.05 -3.64 -3.39 -3.07 -3.13 -3.66 -3.62 -3.35 -3.26 -3.70 -
3.62 -3.57 -3.56 -3.67 -3.14 -2.69

Hypothesis 13:

 -3.44 -2.81 -2.17 -3.23 -2.25 -3.44 -1.55 -3.14 -2.40 -2.45 -2.93 -
3.48 -3.28 -2.66 -2.46 -2.67 -2.62 -2.85 -3.50 -3.04 -2.67 -2.88 -
2.58 -3.11 -2.28 -2.26 -3.22 -3.22 -0.05 -2.99 -3.03 -3.27 -2.29 -
3.09 -3.28 -3.24 -3.04 -3.31 -2.86 -2.62 -3.40 -2.96 -3.35 -3.15 -
3.28 -3.17 -3.19 -3.40 -2.93 -3.17

Hypothesis 14:

 -2.60 -1.11 -1.52 -2.40 -1.64 -3.05 -0.69 -2.10 -0.89 -1.40 -1.99 -
```

```
2.20 -2.71 -1.33 -1.95 -1.35 -2.60 -2.06 -2.42 -2.62 -2.35 -2.71 -
2.00 -2.64 -2.23 -2.49 -2.45 -2.39 -0.73 -2.71 -1.81 -2.72 -1.55 -
2.31 -2.34 -2.77 -1.76 -2.13 -2.56 -2.63 -2.48 -2.40 -2.08 -2.58 -
2.55 -2.41 -2.47 -2.63 -2.56 -2.68

Hypothesis 15:

 -3.04 -1.94 -1.43 -2.95 -1.88 -2.71 -2.37 -2.59 -2.20 -0.33 -1.62 -
2.88 -2.99 -1.93 -1.35 -2.71 -2.61 -2.75 -2.94 -2.87 -2.82 -2.84 -
2.52 -1.62 -2.10 -1.79 -2.85 -2.92 -0.57 -2.95 -2.50 -3.07 -2.14 -
2.81 -2.77 -2.98 -2.68 -2.63 -2.93 -2.90 -2.98 -2.67 -3.03 -2.95 -
3.04 -2.36 -2.90 -3.11 -2.60 -2.98

Hypothesis 16:

 -3.57 -0.58 -2.21 -3.44 -2.38 -3.68 -2.25 -3.13 -2.85 -0.21 -2.14 -
3.19 -3.63 -1.29 -2.12 -3.20 -3.65 -3.19 -3.45 -3.65 -3.42 -3.74 -
2.97 -3.27 -3.13 -3.03 -3.38 -3.53 -1.95 -3.67 -2.88 -3.70 -3.22 -
3.28 -3.33 -3.57 -3.22 -3.21 -3.61 -3.69 -3.32 -3.38 -3.39 -3.69 -
3.63 -3.45 -3.54 -3.51 -3.38 -3.58

Hypothesis 17:

 -3.57 -1.96 -3.00 -3.12 -1.17 -3.86 -2.30 -3.49 -3.31 -2.26 -2.69 -
2.22 -3.45 -2.52 -2.63 -2.23 -3.41 -2.52 -3.82 -2.47 -2.93 -2.72 -
0.09 -2.99 -3.15 -3.48 -2.46 -3.40 -3.23 -3.15 -3.41 -3.66 -3.29 -
2.01 -3.69 -3.21 -3.45 -3.10 -2.50 -3.35 -2.79 -3.52 -3.63 -3.70 -
3.61 -3.60 -3.72 -3.68 -1.42 -3.24

Hypothesis 18:

 -3.71 -2.76 -2.88 -3.16 -0.58 -3.93 -2.55 -3.35 -3.30 -2.35 -3.25 -
3.30 -3.67 -2.26 -3.10 -2.30 -3.53 -1.56 -3.81 -3.48 -3.19 -3.66 -
0.19 -2.81 -2.21 -3.48 -2.35 -2.98 -3.09 -3.51 -3.26 -3.70 -2.93 -
3.31 -3.34 -3.68 -3.49 -2.52 -3.48 -3.49 -3.60 -3.18 -3.55 -3.71 -
3.64 -3.59 -3.62 -3.76 -1.77 -2.90

Hypothesis 19:

 -2.38 -1.77 -1.94 -1.94 -0.88 -2.98 -1.07 -2.03 -1.52 -1.74 -2.24 -
2.30 -2.36 -1.70 -2.09 -1.59 -1.68 -2.12 -2.57 -1.85 -2.01 -1.79 -
1.20 -2.20 -1.80 -2.18 -1.85 -2.07 -1.05 -2.47 -2.33 -2.47 -0.74 -
2.21 -2.42 -2.38 -2.44 -1.28 -1.94 -2.21 -2.49 -2.30 -2.30 -2.15 -
2.37 -2.13 -2.25 -2.55 -2.05 -2.41


Ex[1]: hi, Lex[0]:            hi (-0.17)|

 (* trellis for lexicon word “hi”)
 -2.85 -INF  -INF  -INF  -INF  -0.95 -INF  -INF  -INF  -INF
 -2.89 -3.38 -INF  -INF  -INF  -1.22 -0.99 -INF  -INF  -INF
 -3.70 -3.35 -2.01 -INF  -INF  -1.32 -2.13 -1.12 -INF  -INF
 -2.78 -4.03 -0.04 -0.13 -INF  -0.11 -1.21 -3.06 -1.89 -INF
 -3.23 -3.78 -3.12 -0.05 -0.73 -0.65 -1.10 -0.70 -2.45 -1.40
 -0.57 -1.95 -3.23 -3.09 -1.05 -0.33 -0.21 -2.26 -2.35 -1.74

lexicon word score =  -0.17
….
```

```
(*Trellis for other word lexicon are not shown here*)

Ex[1]: hi, Lex[1]:           Apple (-2.13)|
Ex[1]: hi, Lex[2]:         Between (-INF) |
Ex[1]: hi, Lex[3]:      Capability (-INF) |
Ex[1]: hi, Lex[4]:       Directory (-INF) |
Ex[1]: hi, Lex[5]:           Earth (-1.52)|
Ex[1]: hi, Lex[6]:           Fuzzy (-3.05)|
Ex[1]: hi, Lex[7]:          Giving (-1.87)|
Ex[1]: hi, Lex[8]:        Hydrogen (-INF) |
Ex[1]: hi, Lex[9]:          Island (-2.74)|
Ex[1]: hi, Lex[10]:             Job (-2.20)|
Ex[1]: hi, Lex[11]:    Ku-Klux-Klan (-INF) |
Ex[1]: hi, Lex[12]:         Liberty (-INF) |
Ex[1]: hi, Lex[13]:           Money (-2.35)|
Ex[1]: hi, Lex[14]:           North (-1.43)|
Ex[1]: hi, Lex[15]:         Obvious (-INF) |
Ex[1]: hi, Lex[16]:         Parking (-INF) |
Ex[1]: hi, Lex[17]:            Quiz (-2.23)|
Ex[1]: hi, Lex[18]:          Rabbit (-2.43)|
Ex[1]: hi, Lex[19]:          Smooth (-2.35)|
Ex[1]: hi, Lex[20]:         T-shirt (-INF) |
Ex[1]: hi, Lex[21]:            User (-2.10)|
Ex[1]: hi, Lex[22]:           Voice (-1.88)|
Ex[1]: hi, Lex[23]:       Warehouse (-INF) |
Ex[1]: hi, Lex[24]:           X-ray (-2.25)|
Ex[1]: hi, Lex[25]:          Yuppie (-2.60)|
Ex[1]: hi, Lex[26]:            Zero (-1.77)|

      Top 1 : hi -0.170282
            char : h, start: 0, end: 3
            char : i, start: 4, end: 5

      Top 2 : North -1.432790
            char : N, start: 0, end: 0
            char : o, start: 1, end: 1
            char : r, start: 2, end: 3
            char : t, start: 4, end: 4
            char : h, start: 5, end: 5

      Top 3 : Earth -1.515201
            char : E, start: 0, end: 0
            char : a, start: 1, end: 1
            char : r, start: 2, end: 3
            char : t, start: 4, end: 4
            char : h, start: 5, end: 5

      True label: hi  score -0.170282  position 1

  Recognition time    0.64 seconds
```

Reconnaissance de l'écriture manuscrite en-ligne par approche combinant systèmes à vastes marges et modèles de Markov cachés

Nos travaux concernent la reconnaissance de l'écriture manuscrite qui est l'un des domaines de prédilection pour la reconnaissance des formes et les algorithmes d'apprentissage. Dans le domaine de l'écriture en-ligne, les applications concernent tous les dispositifs de saisie permettant à un usager de communiquer de façon transparente avec les systèmes d'information. Dans ce cadre, nos travaux apportent une contribution pour proposer une nouvelle architecture de reconnaissance de mots manuscrits sans contrainte de style. Celle-ci se situe dans la famille des approches hybrides locale/globale où le paradigme de la segmentation/reconnaissance va se trouver résolu par la complémentarité d'un système de reconnaissance de type discriminant agissant au niveau caractère et d'un système par approche modèle pour superviser le niveau global. Nos choix se sont portés sur des Séparateurs à Vastes Marges (SVM) pour le classifieur de caractères et sur des algorithmes de programmation dynamique, issus d'une modélisation par Modèles de Markov Cachés (HMM). Cette combinaison SVM/HMM est unique dans le domaine de la reconnaissance de l'écriture manuscrite. Des expérimentations ont été menées, d'abord dans un cadre de reconnaissance de caractères isolés puis sur la base IRONOFF de mots cursifs. Elles ont montré la supériorité des approches SVM par rapport aux solutions à bases de réseaux de neurones à convolutions (Time Delay Neural Network) que nous avions développées précédemment, et leur bon comportement en situation de reconnaissance de mots.

Mot-clefs: reconnaissance écriture manuscrite, classifieur, systèmes à vastes marges, modèles de Markov caches, réseau de neurones, programmation dynamique.

On-line Handwriting Recognition using Support Vector Machines and Hidden Markov Models approaches

Handwriting recognition is one of the leading applications of pattern recognition and machine learning. Despite having some limitations, handwriting recognition systems have been used as an input method of many electronic devices and helps in the automation of many manual tasks requiring processing of handwriting images. In general, a handwriting recognition system comprises three functional components; preprocessing, recognition and post-processing. There have been improvements made within each component in the system. However, to further open the avenues of expanding its applications, specific improvements need to be made in the recognition capability of the system. Hidden Markov Model (HMM) has been the dominant methods of recognition in handwriting recognition in offline and online systems. However, the use of Gaussian observation densities in HMM and representational model for word modeling often does not lead to good classification. Hybrid of Neural Network (NN) and HMM later improves word recognition by taking advantage of NN discriminative property and HMM representational capability. However, the use of NN does not optimize recognition capability as the use of Empirical Risk minimization (ERM) principle in its training leads to poor generalization. In this thesis, we focus on improving the recognition capability of a cursive online handwritten word recognition system by using an emerging method in machine learning, the support vector machine (SVM). We first evaluated SVM in isolated character recognition environment using IRONOFF and UNIPEN character databases. SVM, by its use of principle of structural risk minimization (SRM) have allowed simultaneous optimization of representational and discriminative capability of the character recognizer. We finally demonstrate the various practical issues in using SVM within a hybrid setting with HMM. In addition, we tested the hybrid system on the IRONOFF word database and obtained favourable results.

Keywords: handwriting recognition, on-line, support vector machine, hidden markov model, neural network, empirical risk minimization (ERM), structural risk minimization, dynamic programming.